Investigating Entity Knowledge in BERT with Simple Neural End-To-End Entity Linking

Samuel Broscheit
Data and Web Science Group, University of Mannheim, Germany
broscheit@informatik.uni-mannheim.de

Abstract

A typical architecture for end-to-end entity linking systems consists of three steps: mention detection, candidate generation and entity disambiguation. In this study we investigate the following questions: (a) Can all those steps be learned jointly with a model for contextualized text-representations, i.e. BERT (Devlin et al., 2019)? (b) How much entity knowledge is already contained in pretrained BERT? (c) Does additional entity knowledge improve BERT's performance in downstream tasks? To this end, we propose an extreme simplification of the entity linking setup that works surprisingly well: simply cast it as a per token classification over the entire entity vocabulary (over $700K$ classes in our case). We show on an entity linking benchmark that (i) this model improves the entity representations over plain BERT, (ii) that it outperforms entity linking architectures that optimize the tasks separately and (iii) that it only comes second to the current state-of-the-art that does mention detection and entity disambiguation jointly. Additionally, we investigate the usefulness of entity-aware token-representations in the text-understanding benchmark GLUE, as well as the question answering benchmarks SQUAD V2 and SWAG and also the EN-DE WMT14 machine translation benchmark. To our surprise, we find that most of those benchmarks do not benefit from additional entity knowledge, except for a task with very small training data, the RTE task in GLUE, which improves by 2%.

1 Introduction

The goal of entity linking is, given a knowledge base (KB) and unstructured data, e.g. text, to detect mentions of the KB's entities in the unstructured data and link them to the correct KB entry. The entity linking task is typically implemented by the following steps:

- Mention detection (MD): text spans of potential entity mentions are identified,
- Candidate generation (CG): entity candidates for each mention are retrieved from the KB,
- Entity disambiguation (ED): (typically) a mix of useful coreference and coherence features together with a classifier determine the entity link.

Durrett and Klein (2014) were the first to propose jointly modelling MD, CG and ED in a graphical model and could show that each of those steps are interdependent and benefit from a joint objective. Other approaches only model MD and ED jointly (Nguyen et al., 2016; Kolitsas et al., 2018), thus these architectures depend on a CG step after mention detection. Hachey et al. (2013); Guo et al. (2013); Durrett and Klein (2014) showed the influence of CG on entity linking, because it can be the coverage bottleneck, when the correct entity is not contained in the candidates for ED. Yamada et al. (2016, 2017) use a precomputed set of entity candidates published by Pershina et al. (2015) for their experiments on the CoNLL03/AIDA benchmark dataset (Hoffart et al., 2011), and due to this their experiments are comparable across studies with regards to the CG step. MD has a similar impact on entity linking performance, as it determines the upper bound of linkable mentions.

BERT (Devlin et al., 2019) is a deep self-attention-based architecture which is pretrained on large amounts of data with a language modelling objective. This model provides very rich linguistic text-representations that have been shown to be very useful for many NLP tasks. Since its appearance, BERT is being analyzed and applied in various domains (Beltagy et al., 2019; Lee et al., 2019). A recent study found that BERT automatically learns the NLP pipeline (Tenney et al., 2019),

Proceedings of the 23rd Conference on Computational Natural Language Learning, pages 677–685
Hong Kong, China, November 3-4, 2019.

i.e. a stack of increasingly higher level linguistic functions. Zhang et al. (2019) investigated injecting entity knowledge from noisy[1] automatic entity linking into the pretraining of BERT and they could show that this improves relation extraction.

In this study we investigate the following questions:

(a) Can BERT's architecture learn all entity linking steps jointly? We propose an extreme simplification of entity linking and cast it as a per token classification over the entire entity vocabulary, thus solving MD, CG and ED simultaneously (see Fig. 1). The entity vocabulary is based on the $700K$ top most frequent entities in English Wikipedia and the training data was derived from English Wikipedia texts. We first trained BERT-base-uncased on English Wikipedia (dubbed BERT+Entity) and then fine-tuned and evaluated it on an entity linking benchmark. We found that this worked surprisingly well for entity linking, even if we do not have any supervision on mention-spans, i.e. BIO tags. An error analysis with validation data revealed that only 3% of errors are purely due to span errors, while most errors are due to wrong *Nil* predictions which often coincided with entities being infrequent.

(b) How much entity knowledge is already contained in pretrained BERT? To investigate this question, we froze BERT and only trained the entity classifier of BERT+Entity on Wikipedia (dubbed Frozen-BERT+Entity), i.e. the resulting entity classifier is adjusted for entity mentions for which plain BERT already does assign distinct token representations, such that correct entity classification is possible. Then we fine-tuned and evaluated Frozen-BERT+Entity on an entity linking benchmark. We find that the performance of Frozen-BERT+Entity is 6% below BERT+Entity, showing that BERT+Entity has learned additional entity knowledge.

(c) Does additional entity knowledge improve BERT's performance in downstream tasks? Due to training BERT+Entity with a per token classification, the model is forced to assign distinct entity specific features to each token of an entity mention. Downstream tasks could exploit this, if additional entity information is necessary for them. We evaluated BERT+Entity in the natural

[1]TagMe's performance on various benchmark datasets ranges from 37% to 72%. F1 (Kolitsas et al., 2018)

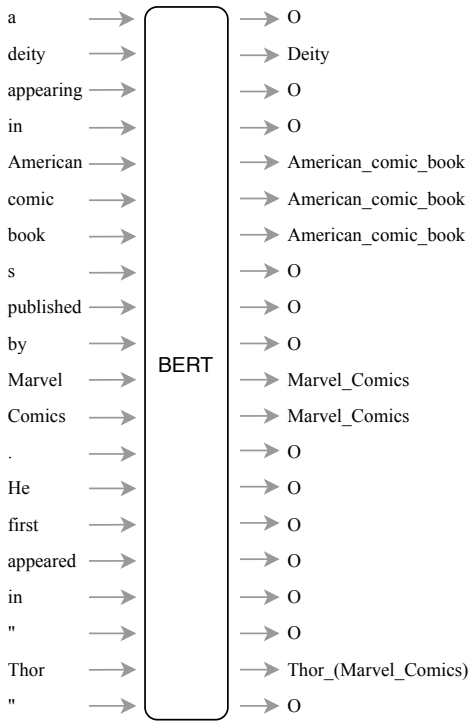

Figure 1: Illustrating the simple neural end-to-end entity linking setup. BERT+Entity predicts entity links per token, where "O" denotes a *Nil* prediction. The example shows how context can help to link *"Thor"* to *Thor_(Marvel_Comics)*.

language understanding benchmark GLUE (Wang et al., 2018), the question answering (QA) benchmarks SQUAD V2 (Rajpurkar et al., 2018) and SWAG (Zellers et al., 2018), and the machine translation benchmark EN-DE WMT14. We confirm the finding from Zhang et al. (2019) that additional entity knowledge is not beneficial for the GLUE benchmark. To our surprise, we also find that additional entity knowledge is neither helpful for the two QA datasets nor for machine translation. The only exception is the RTE task in GLUE in which BERT+Entity improves 2%. This dataset has just 0.5-2% of the training data of the two larger natural language inference datasets in GLUE.

Our contributions are: We are the first to study the latter questions. We are also the first to propose a fully neural model, that does MD, CG and ED all in one model, i.e. performing entity linking without any pipeline or any heuristics. We are also the first to propose to model entity linking as a token classification and show that this seems to be a viable option. We also uncover that there is a lack of tasks that evaluate additional entity knowledge in pretrained language models.

2 Related Work

Entity Linking Durrett and Klein (2014) is the work that is closest to our approach, although not neural. In their approach they model interactions between the MD, CG and ED tasks jointly. They find that the joint objective is beneficial, such that each task improves. They also note that there is no natural order of the tasks and they should interact freely. Their approach to CG is to learn to generate queries to the KB. Nguyen et al. (2016) also propose jointly modelling MD and ED with a graphical model and show that it improves ED performance and is more robust. Kolitsas et al. (2018) recently published their study in which they propose the first neural model to learn MD and ED jointly. Their proposed method is to overgenerate mentions and prune them with a mention-entity dictionary. The ED step reasons over the remaining mentions if and to what they link to. However, modern approaches for solving natural language tasks operate on neural text-representations, and the approaches discussed so far only yield entity-links. Yamada et al. (2016, 2017) was the first to investigate neural text representations and entity linking, but their approach is limited to ED.

Pretrained Language Models ULMFIT (Howard and Ruder, 2018), ELMO (Peters et al., 2018), BERT (Devlin et al., 2019) and GPT-2 (Radford et al., 2019) are modern language models that are very deep and wide (for NLP) and are pretrained on large amounts of data. They provide very rich text representations that have shown to improve many NLP tasks by just replacing the static word embeddings with deep contextualized word embeddings. As Peters et al. (2019) show, further training the deep language models alongside the model that uses the embeddings as input can be helpful, for which the term "finetuning" is used. The current trend in research is to investigate all aspects of these language models, seeking insights in their inner workings (Tenney et al., 2019), or their application to various domains (Beltagy et al., 2019; Lee et al., 2019). In this study, we investigate the factual information in form of entities that is contained in BERT, seeking to understand to what degree this information is already identifiable in BERT and if the entity knowledge can be improved.

3 End-To-End Neural Entity-Linking

In this section we describe the BERT+Entity, which is a is straightforward extension of BERT, however, as with the original BERT, the main challenge lies in designing the training scheme, i.e. in our case the creation of the training data. Our goal for the experiments is to evaluate, if we can learn candidate generation, thus a desiderata is to make the entity vocabulary as large as possible to be comparable to other studies. The text data and the entity linking annotations are derived from Wikipedia by exploiting intra-Wikipedia links. This yields the challenge that the annotations for entity links from Wikipedia are assumed to be incomplete, i.e. not every entity mention in Wikipedia is linked, which we hypothesize can be detrimental during training.

3.1 Model

Our model is based on BERT, which is a deep self-attention-based architecture (Vaswani et al., 2017) that was trained on large amounts of text. Its training objective is two-fold: (a.) predict missing tokens from sentences, and (b.) classify if a second sentence was an adjacent sentence. The input and output token vocabulary are sub-words, i.e. the vocabulary is computed from the training data by determining the $30K$ most frequent character sequences, excluding spaces. Devlin et al. (2019) made several pretrained BERT models publicly available. They differ in size — i.e. token embedding size and self-attention layer depth — and whether the token vocabulary is cased or uncased. BERT+Entity is a straightforward extension on top of BERT, i.e. we initialize BERT with the publicly available weights from the BERT-base-uncased model and add an output classification layer on top of the architecture. Given a contextualized token, the classifier computes the probability of an entity link for each entry in the entity vocabulary. Formally, let d be BERT's token embedding size, and $E \in \mathbb{R}^{|KB| \times d}$ the entity classification layer, with $|KB|$ being the number of entities in the KB, V is the sub-word vocabulary, $c_i = BERT(h)[i]$ is the i-th contextualized token computed by BERT from context $h = [v_1, v_2, ..., v_{i-1}, v_i, v_{i+1}, ..., v_m]$ with each $v \in V$. Consequently, the probability $p(j|v, h)$ of word v — which is the i-th token in context h — linking to entity j is computed by $\sigma(E_j c_i)$, where σ is the sigmoid function.

3.2 Training Data

The entity vocabulary and training data are derived from English Wikipedia texts[2]. We used an extended version of WikiExtractor[3] to extract the text spans that are associated with an internal Wikipedia link to use as annotation, e.g. in the sentence *"The first Thor was all about introducing Asgard"*, the text span *"Thor"* links to *https://en.wikipedia.org/wiki/Thor_(film)*. BERT is originally trained with sentences. However, for entity linking, a larger context can help to disambiguate entity mentions, which is why we select text fragments of such a length, that they span multiple sentences. For later use we collect (m, e) tuples of entities e and their mention m. This yields a set M of potentially linkable strings and also lets us compute the conditional probability $p(e|m)$ based on the $\#(m, e)$ counts.

Handling incomplete annotation A challenge in using the Wikipedia links as annotation is that most entities do not have all their mentions annotated, i.e. often only the first appearance in an article is linked. We hypothesize that learning a classifier on such skewed data would yield a skewed model. Our approach to counter missing annotations is two-fold: (i) We only select text fragments that contain a minimum count of annotated Wikipedia links. (ii) To account for unlinked mentions in the fragments we use a Trie-based matcher[4] to annotate all occurrences of linkable strings that we collected in M. As entity links we annotate all possible entities this mention could link to but only with the conditional probability $p(e|m)$, with the goal that the model remembers a context independent entity prior. One issue is that due to the incomplete annotation, the $\#(e, m)$ counts yield $p(Nil|\text{"}\textit{United States}\text{"}) > 0$, i.e. the mention *"United States"* has a large non-zero probability to link to nothing. Based on the assumption that the mentions of the most popular entities should always link to something, we compute the average of the probability of linking to *Nil* for the $k = 1000$ most frequent entities

$$\bar{p}_{Nil} = \frac{1}{k} \sum_{j} \frac{\#(m_j, Nil)}{\#m_i}.$$

and use $\#(m_i, Nil) - \frac{\bar{p}_{Nil}}{(1-\bar{p}_{Nil})} * \#(m_i, e_*)$ to discount $\#(m_i, Nil)$ such that $p(Nil|\text{"}\textit{United States}\text{"}) \approx 0$, i.e. the model should always link "United States" and mentions of less frequent entities get an increase in probability to link to something.

4 Entity Linking Experiments

In the experiments we want to investigate how the simple neural end-to-end entity linking model BERT+Entity performs, i.e. if it learns something additional on-top of BERT. Additionally, we investigated if the entity-aware token-representations are useful for downstream tasks. We also discuss the main engineering challenges training with such a large entity vocabulary.

4.1 Data

Wikipedia We report two settings which differ in size of the entity vocabulary, size of the fragments and minimum number of entities per fragments. The first setting was the initial study, and the second one is a follow up study in which we changed settings that potentially could improve entity linking performance.

Setting I: We keep the $700K$ top most frequent entities from the $\approx$ $6M$ entities in Wikipedia, i.e. we chose the entity vocabulary as large as it was technically feasible with regards to memory and training speed. To put it into context, the CoNLL03/AIDA entity linking benchmark contains $23,5K$ entities in 1300 documents. We are missing 30 entities from CoNLL03/AIDA that only appear less than 10 times in the Wikipedia training data. We chunk the Wikipedia texts into fragments with a length of 110 tokens and an overlap of 20 tokens with the previous and following fragment. We only keep fragments that contain at least 1 infrequent linked entity or at least 3 frequent ones. This yields $8,8M$ training instances from which we take 1000 each for validation and testing.

Setting II: We keep the $500K$ top most frequent entities, which is comparable to the entity vocabulary of Kolitsas et al. (2018) and we have to add ≈ 1000 entities from CoNLL03/AIDA to the entity vocabulary to be able to evaluate our model on that benchmark. We increase the fragment size to 250 tokens and keep fragments that contain at least 1 linked entity but keep at most 500 fragments per entity. This yields $2,4M$ training instances from which we take 500 each for validation and testing.

[2]From a enwiki Wikipedia dump from 20.06.2017.

[3]https://github.com/samuelbroscheit/wikiextractor-wikimentions

[4]https://github.com/vi3k6i5/flashtext

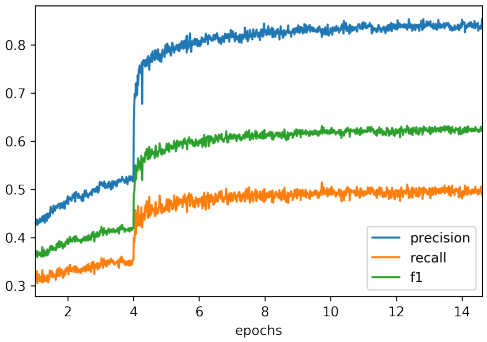

Figure 2: Per token classification InKB scores on the validation data during training on the Wikipedia dataset in Setting II for 40 days. The jump at the 4-th epoch happens when we switch from training Frozen-BERT+Entity to BERT+Entity, i.e. when we start fine-tuning BERT.

Entity Linking Benchmark To evaluate on a commonly used benchmark dataset we use CoNLL03/AIDA. It is the biggest manually annotated ED dataset. It contains 946 documents in training, 216 in validation (testa/AIDA-VALID) and 231 in test (testb/AIDA-TEST).

4.2 Training

We use a multi-class classification over the entity vocabulary, i.e. the label y vector for one token v_i is defined by

$$y_{ij} = p(j|v_i), \text{ for } j \in \{1, .., ||KB||\}.$$

However, computing the loss over the whole entity vocabulary would be infeasible, because the entity mention vocabulary is very large and the gradients for the entity classifier would exceed our GPU memory. Thus, to improve memory efficiency and increase convergence speed, we use negative sampling. After sampling text fragments for a batch b, we collected the set $N{+}_b$ of all true entities — according to the annotations discussed in Sec. 3.2 — that occurred in those text fragments. Ideally we would update the representations of those entities that do not occur in the set $N{+}_b$ which the model is erroneously the most confident about. To achieve this, we first performed a prediction for the text fragments in the current batch and collected for each token the top k predicted entities. We aggregated the entities' logits over the whole batch and sorted the entities by their aggregated logits into the list $N_b{-}$ and removed from it any entity contained in $N_b{+}$. We join $N_b = N_b{+} \cup N_b{-}$ and truncate $N_b{-}$ such that $|N_b|$ equals a given maximum size. Each label vector y_i for token c_i from fragment C in batch b was now defined over the entities in N_b. Thus, we only predict over the corresponding subset of the entity embedding table, i.e. $\hat{E} = E(N_b)$. The loss for one fragment C in batch b was computed by

$$L = \frac{1}{|N_b| * |C|} \sum_{i}^{|C|} \sum_{j}^{|N_b|} -[y_{ij} \cdot \log \sigma(\hat{E}_j c_i) + (1 - y_{ij}) \cdot \log(1 - \sigma(\hat{E}_j c_i))].$$

For training on Wikipedia we used Adam (Kingma and Ba, 2015) with mini batch size 10, gradient accumulation over 4 batches, maximum label size 10240, the learning rate for BERT was $5e$-5 and for the entity classifier 0.01. In Setting I we train the model for 4 epochs, one epoch took five days with two TitanXp/1080Ti. In the first 1.5 epochs we train Frozen-BERT+Entity and then BERT+Entity. In Setting II we train the model for 14 epochs and one epoch took three days. In the first 3 epochs we train Frozen-BERT+Entity and then BERT+Entity.

For training on CoNLL03/AIDA we used Adam (Kingma and Ba, 2015) with mini batch size 10, gradient accumulation over 4 batches, maximum label size 1024, learning rates for BERT $5e$-5, dropout in BERT 0.2, and we freeze the token embeddings, the first two layers of BERT and the entity classifier. We train the remaining parameters for up to 30 epochs and perform early stopping according to strong match (see next Section). One epoch took seven minutes with one TITAN Xp/1080 Ti.

4.3 Performance Metrics

We compute the Micro InKB Precision, Recall and F1 metrics and we only consider entities as true, if they are in our KB. We compute a strong match, i.e. every token in the gold annotated span has to be classified correctly. We also report a weak match, which we define as at least one token in the gold annotated span having to link to the correct entity. This setting accounts for annotation inconsistencies, e.g. when the model and the annotation do not agree on which mention *"U.S. army"* or *"U.S."* to annotate (can be either way). We also report strong ED Precision@1, i.e. we ignore *Nil* predictions of the model and only evaluate the top ranked entity only for spans that have a gold entity.

		AIDA/testa			AIDA/testb		
		strong F1	weak F1	ED	strong F1	weak F1	ED
Kolitsas et al. (2018) indep. baseline		75.7	76.0	-	73.3	73.9	-
Kolitsas et al. (2018)		86.6	87.2	92.4	82.6	83.2	89.1
BERT		63.3	66.6	67.6	49.6	52.4	52.8
Setting I	Frozen-BERT+Entity	76.8	79.6	80.6	64.7	68.0	68.6
	BERT+Entity	82.8	84.4	86.6	74.8	76.5	78.8
Setting II	Frozen-BERT+Entity	76.5	80.1	79.6	67.8	71.9	67.8
	BERT+Entity	86.0	87.3	92.3	79.3	81.1	87.9

Table 1: Comparing entity linking results on CoNLL03/AIDA. *strong F1* and *weak F1* denote InKB F1 scores. *ED* is Precision@1 for InKB. Kolitsas et al. (2018) also study a neural model, however, they only model MD and ED. The independent baseline shows how their model performs when they use mentions detected by Stanford NLP. In Frozen-BERT+Entity BERT is not trained and only the entity classifier on-top is trained.

4.3.1 Results

In Table 1 we compare our results to the most recent results by Kolitsas et al. (2018) who studied a neural approach that does joint modelling of MD and ED, but not CG. They also provide a baseline in which they show how their classifier performs when MD and ED are independent, i.e. linking mentions detected by Stanford NLP.

For the reported results denoted only with BERT, the entity classifier is trained from scratch on CoNLL03/AIDA and BERT is finetuned. This shows the lower bound on this dataset, i.e. the amount of information that we can learn with BERT only from the CoNLL03/AIDA training data. Note, that this cannot generalize to entities that are not contained in training. The difference between BERT and Frozen-BERT+Entity shows the amount of entity knowledge that plain BERT already had, which it transferred in the entity classifier during training on Wikipedia. Finally, BERT+Entity is the proposed model, in which both BERT and the entity classifier have been trained on Wikipedia.

4.3.2 Discussion

Comparing BERT+Entity and Frozen-BERT+Entity we see that there is a significant amount of entity knowledge that BERT+Entity learns additionally to Frozen-BERT+Entity, i.e. training BERT+Entity increases the scores between 6%-10% depending on the score and dataset. However, it should also be noted that Frozen-BERT+Entity already shows an increase of 13%-16% over BERT, thus it already learns for many entities distinct features that enable the entity classifier to identify them. The improvement of Frozen-BERT+Entity in contrast to BERT on CoNLL03/AIDA shows that this pretraining generalizes to validation and test data. We can also observe that Setting II improves by a large margin over Setting I and comes very close to the results of Kolitsas et al. (2018). We conjecture that the biggest impact on the performance from changing the training from Setting I to Setting II, was due to the downsampling of the training data in favor of less frequent entities. This reduction of training data in Setting II — caused by capping the maximum amount of examples per entity — enabled us to run more epochs in less time, which might have improved the representations of less frequent entities.

Reason for error	#
no prediction	57
different than gold annotation	
no obvious reason	13
semantic close	4
lexical overlap	5
nested entity	5
gold annotation wrong	12
span error	3
unclear	1
	100

Table 2: Investigating the types of strong precision errors of BERT+Entity trained in Setting I on CoNLL03/AIDA (testa) on 100 randomly sampled strong precision errors from the validation dataset.

Task	Metric	BERT-BERT-Ensemble	BERT+Entity-Ensemble
CoLA	Matthew's corr.	59.92	59.97
SST-2	accuracy	92.73	92.43
MRPC	F1/accuracy	89.16	90.13
STS-B	Pearson/Spearman corr.	89.90	89.60
QQP	accuracy	91.64	91.21
MNLI	matched acc./mismatched acc.	84.96	84.78
QNLI	accuracy	91.21	91.15
RTE	accuracy	71.48	73.64
WNLI	accuracy	56.33	56.33
SQUAD V2	matched/mismatched	76.89/73.83	76.36/73.46
SWAG	accurracy	80.70	80.76
WMT14 EN-DE	BLEU	22.51	22.20

Table 3: Experiments on downstream tasks with BERT+Entity trained in Setting I. The first group are the GLUE tasks, then followed by SQUAD V2 and SWAG (for which only the dev set results are reported), and the results for machine translation WMT14 EN-DE.

When we compare BERT+Entity with the two results from Kolitsas et al. (2018), we observe that BERT+Entity improves over the baseline that models MD, CG and ED independently, and that BERT+Entity comes second to the current state-of-the-art in end-to-end entity linking. What can also be observed is that the performance of all models drops from AIDA/testa to AIDA/testb. For BERT+Entity, however, the drop is more severe, obviously the model overfits to some patterns in the training data that are present in the validation data, but not in the test data. We hypothesize that this might be due to some sport specific documents that make roughly $1/4$ of the dataset's mentions. However, without spoiling the test-set we cannot know for sure.

In Table 2 we performed an error analysis for the experiments for Setting I to learn what kind of strong precision errors are responsible for the performance of BERT+Entity. The largest source of errors was that BERT+Entity did predict *Nil* instead of an entity. We hypothesized that most of the *no prediction* errors are because those entities have only a low frequency in the training data, i.e. this could be solved by increasing the model size and improving the training time. Another source of error we observed was that the context size was too small due to the fragment size. A surprisingly positive result from the error analysis was that in only 3% a wrong span caused the error. Motivated by the observations we devised the follow-up experiment Setting II (see Section 4.1) in which we changed some of the settings to potentially solve the observed issues.

5 Downstream Tasks Experiments

In this section we discuss the downstream task results. We performed evaluations on the natural language understand task GLUE, the question answering tasks SQUAD V2 and SWAG and the machine translation benchmark EN-DE WMT14. We found that only in one of the subtasks of GLUE —the natural language inference tasks RTE— BERT+Entity performs better than BERT, for all other we can observe no such effect. The reported results are for Setting I, however, we repeated the experiments with Setting II and observed the same outcomes.

5.1 Model

For the tasks GLUE, SQUAD V2 and SWAG we extend hugginface's implementation[5] and concatenate the outputs of BERT and BERT+Entity (dubbed BERT+Entity-Ensemble) or two BERTs (dubbed BERT-BERT-Ensemble). For EN-DE WMT14 we use BERT (dubbed BERT-2Seq) or BERT+Entity (dubbed BERT+Entity-2Seq) as encoder and use a Transformer decoder by adapting fairseqs Pytorch Seq2Seq Transformer implementation (Ott et al., 2019).

[5]https://github.com/huggingface/pytorch-pretrained-BERT

5.2 Training

For the GLUE benchmark, SQUAD and SWAG we train the BERT+Entity-Ensemble and BERT-BERT-Ensemble for 3 epochs and use the default hyper-parameters from the implementation. The models BERT-2Seq and BERT+Entity-2Seq we train for 4 epochs, with Adam as optimizer and learning rate $5e$-5, max 1000 tokens per batch, clip gradient norm 0.1, dropout 0.2, label smoothing 0.1, and we keep the encoders BERT and BERT+Entity fixed for the first epoch and then train it together with the decoder.

5.3 Results

We find that the additional entity knowledge is not helpful in the evaluated tasks. The results in Table 3 show that, except for RTE, there seems to be no advantage in having additional entity knowledge. The question is, if this is (a) due to the entity overlap in training and testing such that also an entity unaware model can learn the necessary model, or (b) the entities are too scarce in the training data to make a difference, or (c) the tasks themselves do not require entity knowledge, i.e. other textual cues are enough. We leave those questions for future research.

6 Conclusion

In this study we investigated an extremely simplified approach to entity linking that worked surprisingly well and allowed us to investigate entity knowledge in BERT. Even when there is a gap to the current state-of-the-art in entity linking, we hypothesize that this gap can be closed with larger hardware capacity to scale up the model size and effective training time. Apart from that, the model is the first that performs entity linking without any pipeline or any heuristics, compared to all prior approaches. We found that with our approach we can learn additional entity knowledge in BERT that helps in entity linking. However, we also found that almost none of the downstream tasks really required entity knowledge, which is an interesting observation and an open question for future research.

Acknowledgments

The author would like to gratefully thank the NVIDIA corporation for the donation of a TITAN Xp GPU that was used in this research.

References

Iz Beltagy, Arman Cohan, and Kyle Lo. 2019. Scibert: Pretrained contextualized embeddings for scientific text. *CoRR*, abs/1903.10676.

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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Greg Durrett and Dan Klein. 2014. A joint model for entity analysis: Coreference, typing, and linking. *TACL*, 2:477–490.

Yuhang Guo, Bing Qin, Yuqin Li, Ting Liu, and Sheng Li. 2013. Improving candidate generation for entity linking. In *Natural Language Processing and Information Systems - 18th International Conference on Applications of Natural Language to Information Systems, NLDB 2013, Salford, UK, June 19-21, 2013. Proceedings*, pages 225–236.

Ben Hachey, Will Radford, Joel Nothman, Matthew Honnibal, and James R. Curran. 2013. Evaluating entity linking with wikipedia. *Artif. Intell.*, 194:130–150.

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 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 782–792.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 328–339.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

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.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So,

and Jaewoo Kang. 2019. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *CoRR*, abs/1901.08746.

Dat Ba Nguyen, Martin Theobald, and Gerhard Weikum. 2016. J-NERD: joint named entity recognition and disambiguation with rich linguistic features. *TACL*, 4:215–229.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Maria Pershina, Yifan He, and Ralph Grishman. 2015. Personalized page rank for named entity disambiguation. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 238–243.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 2227–2237.

Matthew E. Peters, Sebastian Ruder, and Noah A. Smith. 2019. To tune or not to tune? adapting pretrained representations to diverse tasks. In *Proceedings of the 4th Workshop on Representation Learning for NLP, RepL4NLP@ACL 2019, Florence, Italy, August 2, 2019.*, pages 7–14.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 784–789.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4593–4601.

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: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 6000–6010.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the Workshop: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@EMNLP 2018, Brussels, Belgium, November 1, 2018*, pages 353–355.

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, CoNLL 2016, Berlin, Germany, August 11-12, 2016*, pages 250–259.

Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. 2017. Learning distributed representations of texts and entities from knowledge base. *TACL*, 5:397–411.

Rowan Zellers, Yonatan Bisk, Roy Schwartz, and Yejin Choi. 2018. SWAG: A large-scale adversarial dataset for grounded commonsense inference. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 93–104.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: enhanced language representation with informative entities. *to appear*.