UNIVERSITÄT PADERBORN
Die Universität der Informationsgesellschaft
Faculty for Computer Science, Electrical Engineering and Mathematics
Department of Computer Science
Research Group Data Science

Seminar Report
Submitted to the Data Science Research Group
in Partial Fullfilment of the Requirements for the Degree of
Master of Science

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

by
Siddhanth Janadri

Seminar Supervisor:
Dr. Mohamed Sherif

Paderborn, February 7, 2021

Abstract.
In an information extraction pipeline, there involves three steps: Named Entity Recognition which is also known as mention detection (NER), candidate generation (CD) and Named Entity Disambiguation (NED). The process of performing these tasks jointly is known as end-to-end entity linking. Many researchers have been successful in developing models that perform mention detection (NER) and named entity disambiguation (NED) jointly. But this paper deals with things that answers couple of related questions. We try to investigate whether a model can learn all the three steps: NER, CD and NED jointly. This paper also tells the amount of entity knowledge that is present in pre-trained BERT. The possibility of improving the performance of BERT model by providing additional entity knowledge is also checked. In simple terms, these questions have been answered by performing per token classification where every token is checked over the entire entity vocabulary. It has been found that the most of the downstream tasks aren't benefited by the additional entity knowledge.

Contents

0.1 Introduction

Entity linking is the process of assigning textual mentions of entities to the corresponding entries in a knowledge base. Entity linking is implemented by three steps namely,

Named Entity Recognition: detecting all the words in text that refer to words of interest called entities. These entities can be places, people, organizations etc.

Candidate Selection: generating a list of candidates for the identified entities from any knowledge base eg: DBpedia.

Named Entity Disambiguation: disambiguating all the candidates generated from the knowledge base (KB) that are irrelevant to the identified entities and linking them to a unique identifier in a KB.

The joint process of first analyzing and annotating entities within a text and then performing entity disambiguation is known as End-to-End Entity Linking. Figure 1 illustrates the process of entity linking. Named Entity Recognition identifies all the entities from the input text. Following this, candidate selection process produces the list of entities that are relevant to the identified entities in the previous process. Finally, Named Entity Disambiguation process find the most probable or suitable entity from the candidate list and links it to knowledge base by determining the context of text. It makes use of word embeddings where each entity is represented as a vector of real numbers and the words with similar meanings or all the related words with respect to context are placed close to each other in vector space.

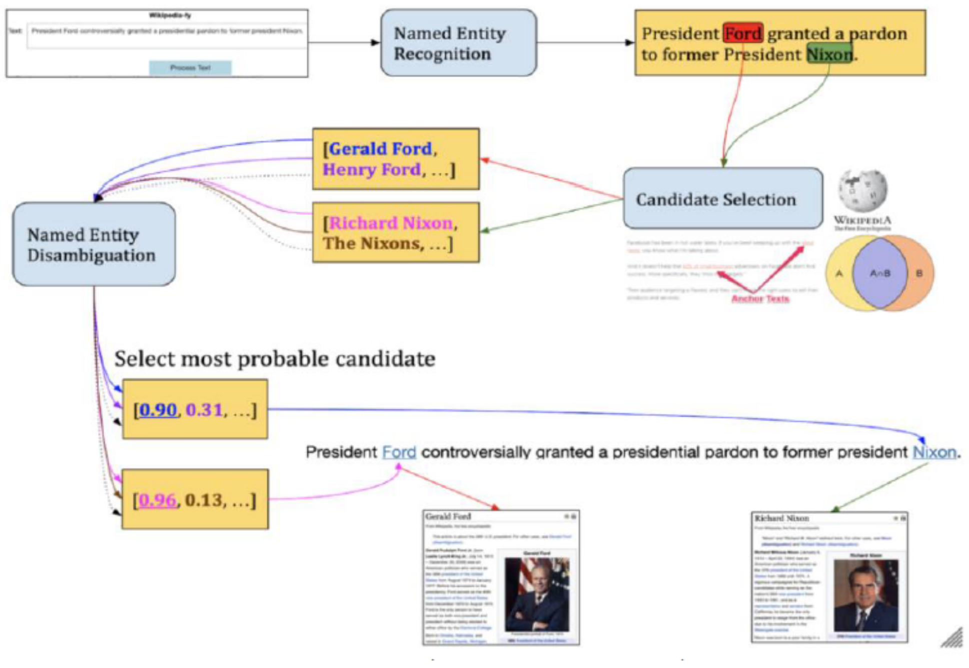

Named Entity Recognition
President Ford granted a pardon to former President Nixon.
[Gerald Ford, Henry Ford, ...]
[Richard Nixon, The Nixons, ...]
Candidate Selection
Named Entity Disambiguation
Select most probable candidate
[0.90, 0.31, ...]
[0.96, 0.13, ...]
President Ford controversially granted a presidential pardon to former president Nixon.
Anchor Texts
Gerald Ford
Richard Nixon

Figure 1: Overview of Entity Linking[1]

[1]https://towardsdatascience.com/named-entity-disambiguation-boosted-with-knowledge-graphs-4a93a94381ef

Essentially, as an introduction, BERT stands for Bidirectional Encoder Representations from Transformers that is pre-trained from unlabelled text corpus which learns contextual relations between words and sentences in a text with the help of attention mechanism. BERT has been very instrumental in various NLP tasks and domains due to its pre-trained nature which requires some tweak to make it a purpose-specific model. This tweak could be few changes in the architecture or the addition of new layers that serve as input or output layers.

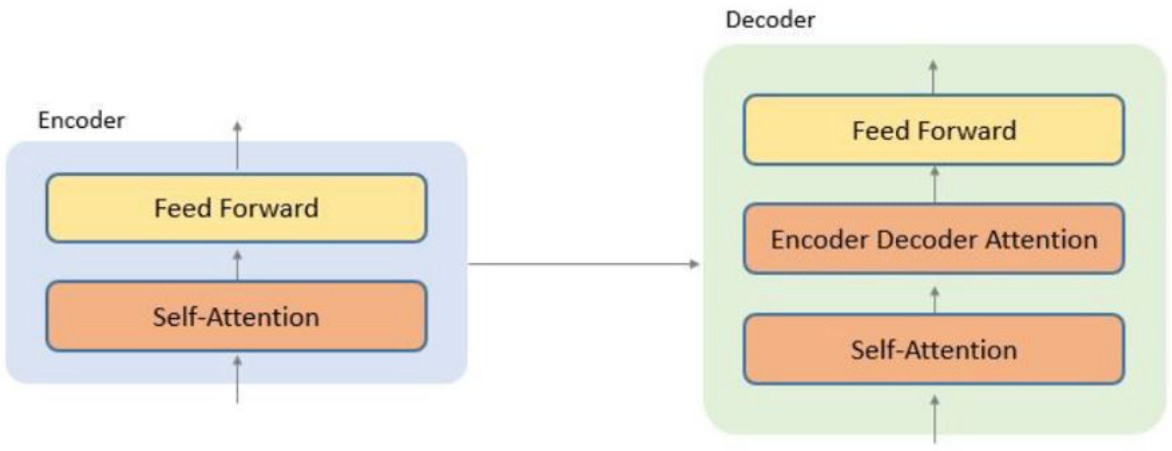


Figure 2: BERT model architecture[2]

BERT is basically a language model with the stack of encoders and decoders on top of one another. BERT holds the upper hand because of its ability to read text input sequentially in both directions simultaneously i.e. left to right and right to left. As previous models could only read the text either from left to right or from right to left, BERT holds a special place among pre-trained language models. It has been pre-trained on two prediction tasks[1]:

Masked Language Modelling: The input text that is fed to the model will have 15% of masked tokens. The objective of this task is to predict those masked tokens by understanding the context of input sequence. In other words, the model has to predict unmasked tokens with the help of masked ones.

Next Sentence Prediction: Given the sentence pairs as input to the model, the goal is to predict the relationship between the two sentences. It has to predict if the second sentence is an adjacent sentence in the text document. It is trained with just 50% of the sentence pairs that are adjacent to each other in the actual document.

BERT models are available with many variants that differ by token embedding size and encoder-decoder layers depth. Two such variants are $\text{BERT}_{\text{base}}$ with 12 transformer blocks and $\text{BERT}_{\text{large}}$ with 24 transformer blocks. Throughout this paper, BERT-base-uncased model is chosen for fine-tuning and for experimental purposes.

Durrett and Klein[2] have come up with a way to model all the three steps in a graphical way that could show that these steps are in-fact interdependent. Kolitsas and Ganea[3] have

[2]https://towardsdatascience.com/what-is-attention-mechanism-can-i-have-your-attention-please-3333637f2eac

modelled an approach that performs NER and NED steps jointly by leveraging the concept of coreference resolution. This is in-fact a neural approach to jointly detect the entities in the document and link those entities to appropriate knowledge base entities. Nguyen, Theobald and Weikum[4] have developed a probabilistic graphical model to perform NER and NED steps jointly that considers mention spans, its types and entity links that show the mapping of mentions to knowledge base entities[4]. There can be cases where required entities aren't available in the candidate list that in-turn hinders the performance due to lack of coverage. Few researchers along with Durrett and Klein brought out the importance of candidate generation step and its importance in solving any NLP problem. Relation extraction is the task of predicting attributes and relations for entities in a sentence. It also extracts semantic relationships from a text. Zhang et al.[5]. has found that relation extraction can be improved by entity knowledge that is derived from noisy entity linking.

0.2 Problem Definition

Three main questions are addressed related to end-to-end entity linking and fine-tuned BERT[6]. First question is whether BERT can learn all entity linking steps jointly. This has been dealt using per token classification over entire entity vocabulary. The entity vocabulary is derived from 700k top-most frequent entities from English Wikipedia. Per-token classification is a way of predicting entity links for every token in a sentence or a text fragment based on the context. Initially, just the pre-trained model of BERT – BERT-base-uncased model is trained followed by training fine-tuned model. Later, the fine-tuned BERT-base-uncased architecture model is compared with baselines of entity linking to evaluate it. Results of this have been pretty impressive even without BIO tags used for supervision. BIO format[3] is a tagging format for tagging tokens in a chunk which represents the beginning of a chunk(B), internal of a chunk(I) and 'O' represents that the token doesn't belong to any chunk.

The next question deals with the amount of entity knowledge that is present in BERT[6]. This has been evaluated by training Frozen-BERT+Entity model which means that only classification layer of BERT+Entity model is trained by gorgonizing BERT. As a result, two models – BERT+Entity and Frozen-BERT+Entity are available where both BERT and classification layer are trained in former model and only classification layer is trained in latter model. Therefore, the amount of entity knowledge that is present in plain-BERT can be determined by the difference of these two models. This makes it possible to show that entity knowledge is already available in BERT.

The last question is all about improving performance by additional entity knowledge. The extra distinctive features are assigned to each token of an entity mention in order to increase the knowledge of entity. It is imperative to know that none of the baselines of language understanding tasks, QA datasets and machine translation task are benefited by the additional entity knowledge. But there is an exception in natural language understanding benchmark GLUE where one of its tasks called RTE is benefitted by additional entity knowledge. RTE stands for Recognized Textual Entailment[1] is a task where the second sentence in the sentence pair has to predict whether its relation with the first one is either entailment, contradiction or neutral.

[3]https://medium.com/analytics-vidhya/bio-tagged-text-to-original-text-99b05da6664

0.3 Related Work

End-to-End Entity Linking

The process of recognizing entity mentions in text and linking them to corresponding entries in a knowledge base is called as Entity Linking. However, in order to perform this, three steps are carried out namely, Named Entity Recognition (NER), Candidate generation and Named Entity Disambiguation (NED). The joint process of performing all these tasks as a single task leveraging mutual dependency is known as End-to-End Entity Linking. There are few challenges with respect to joint modelling of these tasks. Sil and Yates et al.[7] have tackled the problem where second and third steps of entity linking recover the errors performed by NER step. They have modelled in such a way that it generates more mentions and lets the NER step to take the final decisions.

Kolitsas et al.[3] is the first to develop a neural model to perform NER and NED jointly. The approach first considers all possible mentions in the document and learns contextual similarity measures over the list of entity candidates. Apparently, their method chooses only those text fragments that have at least one entity candidate. Based on this, context aware compatibility score is provided to each mention-entity pair which is then integrated with neural attention. The eligible mentions and candidates are then pruned by the process of global voting. It has been identified that mention boundaries are important to distinguish between weak matching EL and strong matching EL. This model would achieve the best results among models that perform NER and NED jointly if the training data is as accurate and consistent as that of test data used for evaluation purpose.

Overall, Durrett and Klein[2] are the first to develop a model that performs end to end entity linking. They have addressed it with a two-level approach where Unary factors are added to carry local features of every task and higher-order factors are added to encode interactions between tasks. Ultimately, they could show that these steps are indeed interdependent. This model shows a huge improvement in performance when compared to baselines that model all the entity linking steps individually.

All the above methods consider only entity links to link entities to knowledge base. Text representations can also be utilized for the same purpose with the help of word embedding methods. Each of the entity is represented as a vector of real numbers and they are placed in such a way that the words of similar meanings are close to each other in vector space. Yamada et al.[8] is the first to consider both text representations and entity links to perform entity disambiguation. This model maps words and entities to each other and they are placed in the vector space such that similar words and entities are close to each other. This similarity is evaluated by computing its cosine similarity.

Pretrained Language models

A lot of neural networks such as recurrent neural networks, convolutional neural networks are being used to solve major natural language processing tasks[9]. The syntactic and semantic features of a language are represented by low-dimensional and dense vectors which makes it easy to develop NLP systems using neural methods. But the drawback of using neural networks is that they don't generalize well on the training data as these networks usually have a lot of parameters. Even the generation of labelled text dataset is very expensive and a challenging task. Therefore we need such models that gets trained on unlabeled text corpora which is relatively easier to generate. Pre-training is done on such large unlabeled datasets and the models understand the universal representations of the language. Such models are called as pre-trained models and by using such models, any NLP tasks can be solved without having to train the model from scratch.

There are a lot of models that are pre-trained on large amount of data such as ELMO (Peters et al.[10]), FIT (Howard and Ruder[11]) and BERT (Devlin et al.[1], 2019). Peters et al.show that by changing appropriate inputs and outputs or by adding new layers on top of the existing pre-trained model, the performance of the overall model along with changes show a lot of improvement which is known as fine-tuning. Fine-tuning a language model can be done according to the task that is to be solved. In this paper, BERT model is fine-tuned in such a way that it performs all the tasks of entity linking jointly. The amount of knowledge that is already contained in BERT is also investigated in terms of entities.

Input embeddings

There is a limitation of using the transformer model where the input text has to be divided into number of segments that results in context fragmentation[12]. This is where BERT is better than transformer model. Every input to the BERT model is a combination of three embeddings.

Position embeddings: These are the embeddings that indicate the position of words in a sentence. This embedding overcomes the problem of transformer where it fails to capture the order of a sentence due to fragmentation.
Segment Embeddings: As we know, BERT is pre-trained with Next sentence prediction task and the sentence pairs have to be fed as input for this task. This embedding indicates the sentence number in the pair that helps the model to distinguish between them.
Token embeddings: Every sequence is broken down to form individual words called tokens. Every token is converted into vector representations so that each token will have some value associated with it.

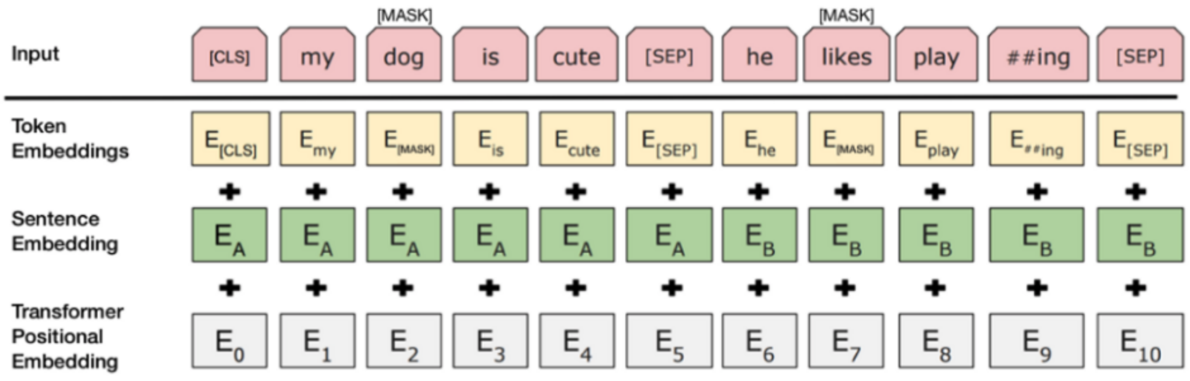

Input
[CLS]
my
[MASK]
dog
is
cute
[SEP]
he
[MASK]
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} $E_{[MASK]}$ E_{is} E_{cute} $E_{[SEP]}$ E_{he} $E_{[MASK]}$ E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Sentence Embedding
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Transformer Positional Embedding
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

Figure 3: Input embeddings in BERT[4]

[4]https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270

0.4 Approach

0.4.1 Model Overview

As specified before, BERT model can be fine-tuned to make it work as a purpose-specific model. The concept of transfer learning can be leveraged by making some changes to the existing pre-trained model so that it can be used for wide variety of tasks. BERT+Entity is a model that has been developed in this approach. It is an extension of BERT model with an additional output classification layer on top of it. The role of the classification layer is to predict the entity link by computing the probability of finding the link in the entity vocabulary. Therefore, BERT+Entity is just a model that has an additional output classification layer on top of BERT's architecture along with random weights that are used for initialization. This has a goal of learning candidate generation that is needed to make the vocabulary big enough so that it can be used in other experiments and studies. Every mention in the text is not linked to knowledge base which becomes apparent that the annotations for entity links are incomplete. Figure 4.1 shows how BERT+Entity model is linking Thor to Thor(Marvel_Comics) based on the context. O indicates that nothing is predicted for that particular token.

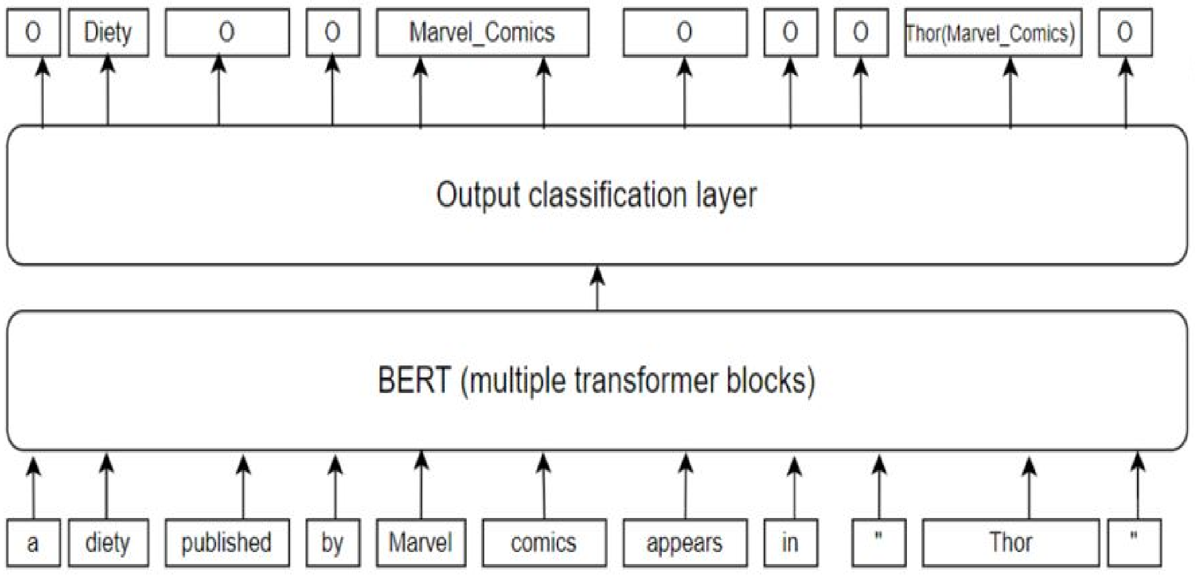


Figure 4: BERT+Entity model

0.4.2 Implementation

The principle of per token classification is used for entity linking and disambiguation. As per this principle, each token is individually checked against the entire entity vocabulary to check for appropriate links. Given the language model BERT which is already pre-trained, BERT+Entity model understands the context of the text fragment with the help of attention mechanism. By leveraging all these concepts and principles, every token in the text fragment will link to something based on the context. If the token is trivial or not of much importance whose absence doesn't have much impact in the text fragment, then the model predicts nothing for such tokens e.g. prepositions.

Formalisation

The entity classification layer is denoted by $E \in R|KB| * d$ where $|KB|$ denotes the number of entities in KB and d denotes the token's embedding size. The contextualized token $c_i = BERT(h)[i]$ is computed by BERT from context $h = [v_1, v_2, \ldots..v_{i-1}, v_i, v_{i+1} \ldots .v_m]$ with each $v \in V$ where V is the sub-word vocabulary. The probability of entity link for each entry in the entire vocabulary is given by $p(j|v, h)$ where word v is the i-th token in context h. This probability is calculated by $\sigma(E_j c_i)$ where σ is a sigmoid function[6].

0.4.3 Incomplete annotation problem

The entity vocabulary that consists of 700k top most frequent entities are derived from English Wikipedia. As BERT is trained with the help of sentences, bigger text fragments that consist of multiple sentences are preferred for experimental purposes. The reason to choose such text fragments is to make the entity linking perform better which ultimately improves the performance of the model. Entity linking will be performed better with bigger text fragments that span multiple sentences because the model gets enough sentences to understand the context which ultimately helps in predicting the links to appropriate entities. But there is a limitation of using Wikipedia links as annotation due to its incomplete annotation problem. It has been found that only the first occurrence of entities' mentions are annotated. Therefore, mentions of most of the entities aren't annotated. Popular entities are those who have all their mentions annotated but there exists some entities where only some of their mentions are annotated which are unpopular entities. Therefore, to handle such a problem, we choose those text fragments that have less annotated Wikipedia links followed by Trie-based matcher which is used for annotating all occurrences of entities' mentions that are collected as linkable strings in set M. This set M consists of (m, e) tuples of e entities and their mentions m to calculate the conditional probability $p(e|m)$.

Due to incomplete annotation, mentions of less frequent entities have a non-zero probability to link to nothing. But there is an entity linking assumption where mentions of all the frequent and popular entities should be linked. This gives us a probability relation for non-frequent entity mentions, e.g. $p(Nil|\text{"Christopher Nolan"}) > 0$ which means that the mention 'Christopher Nolan' has the probability of more than zero to link to nothing. Therefore, we calculate the average of the 'linking to Nil' probability for k=1000 most frequent entities as given below.

$$\bar{p}Nil = \frac{1}{k} * \sum_j \frac{\#(m_j, Nil)}{\#m_i} \tag{1}$$

Later, we use the below formula so that the mentions of less frequent entities will have some probability greater than zero to link to something.

$$\#(m_i, Nil) - \frac{\bar{p}Nil}{1 - \bar{p}Nil} * \#(m_i, e_*) \tag{2}$$

0.5 Experiments

The motivation of these experiments is to check the performance of BERT+Entity and to check if it learns something additional than BERT. There are two settings that have been set to perform training for model evaluation. These settings differ by the type of dataset, number of total entities that are present in the dataset, fragment size, total training instances and so on. The setting 1 is configured as part of initial study to check how the model performs on the simple data with minimized constraints. By evaluating the results of setting 1, setting 2 has been configured in such a way that the performance of the model and the entity linking improves. Figure 5 shows different specifications of these settings.

Setting 1:

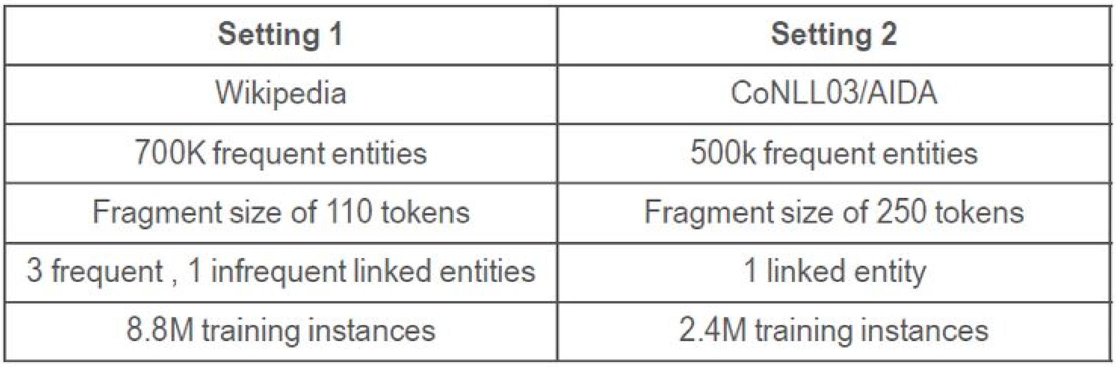
700k most frequent entities are chosen over approximately 6M entities in Wikipedia for the experiment. The Wikipedia text is divided into numerous fragments with 110 tokens each. We can expect an overlap of 20 tokens with immediate neighbour fragments. We choose only those fragments that have at-least 1 rarely occurring linked entity or 3 frequently occurring ones. This results in 8.8M instances for training where 1000 each will be used for validation and testing.

Setting 2:

500k most frequent entities are added along with approximately 1000 entities from CoNLL03/AIDA dataset to compare our model with the benchmark. In this setting, the size of the fragment is increased to 250 tokens with each fragment containing at-least 1 linked entity. This results in 2.4M instances from which 500 each will be taken for validation and testing. The CoNLL03/AIDA is used because this is the biggest annotated entity disambiguation dataset that contains a total of 946 documents that are used for training[6].

0.5.1 Training

The label y vector for one token v_i is given by, $y_{ij} = p(j|v_i), for j \in 1, .., |KB|$. As it is not viable to calculate loss over the entire entity vocabulary, negative sampling is used to improve memory efficiency. Every batch b has a number of text fragments and the set N_b+ with annotated entities are collected and all those entities that are not in set N_b+ are updated with representations. Therefore, a prediction for text fragments is done in every batch and top k predicted entities for each token are collected. Later, the set N_b- is used to add all the aggregated entities' logits over the entire batch and those entities that are present in both the sets are removed from N_b-. We

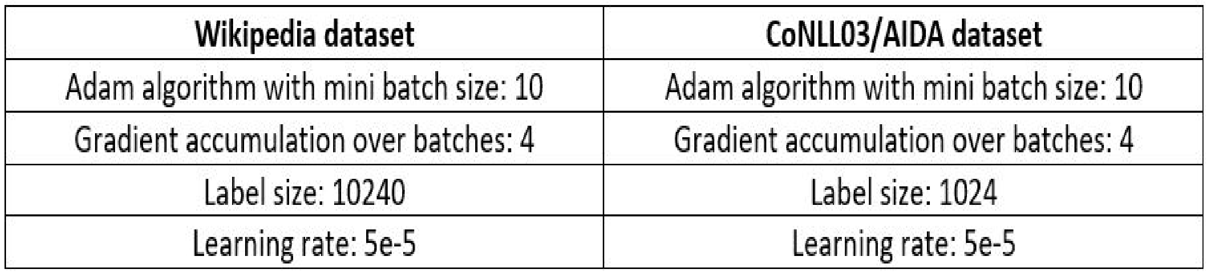


Setting 1	Setting 2
Wikipedia	CoNLL03/AIDA
700K frequent entities	500k frequent entities
Fragment size of 110 tokens	Fragment size of 250 tokens
3 frequent , 1 infrequent linked entities	1 linked entity
8.8M training instances	2.4M training instances

Figure 5: Settings for training

later join $N = N_b + \cup N_b-$ and truncate N_b- such that $|N_b|$ equals the maximum size. Every batch b has numerous fragments C where each label vector y_i for token c_i is defined over the entities in N_b. This makes it possible to use the subset of the entity embedding table from where the predictions of fragments can be done[6].

The training is done on Wikipedia and CoNLL03/AIDA datasets with same mini batch size and different maximum label size. The gradient accumulation is done over 4 batches for both the datasets and learning rate is kept the same. We train with different epochs in both the settings which gives us variety of results to compare with benchmarks. On Wikipedia dataset, the model has been trained for 4 epochs in setting 1 out of which Frozen-BERT+Entity has been trained for the first 1.5 epochs followed by BERT+Entity for the remaining epochs. Similarly a total of 14 epochs has been considered for training the model in setting 2 in which first 3 epochs has been used for training Frozen-BERT+Entity followed by BERT+Entity for the remaining epochs. Figure 6 shows the parameter specifications of both the datasets.



Wikipedia dataset	CoNLL03/AIDA dataset
Adam algorithm with mini batch size: 10	Adam algorithm with mini batch size: 10
Gradient accumulation over batches: 4	Gradient accumulation over batches: 4
Label size: 10240	Label size: 1024
Learning rate: 5e-5	Learning rate: 5e-5

Figure 6: Training parameters

Performance and its metrics

A lot of metrics have been used to measure the performance of the model. Micro InKB Precision determines how many named entities have been disambiguated correctly in the text corpus with the condition of having only mentions of valid knowledge base entities used for validation. Even Recall and F1 metrics have been used to compare and evaluate results with the benchmarks. All the entities that are present in the knowledge base are considered to determine the number of tokens that are predicted properly or linked to the knowledge base in the gold annotated span which is known as strong match. In order to report ED Precision@1, only those entities that are linked to something are considered ignoring all the tokens that are predicted to Nil.

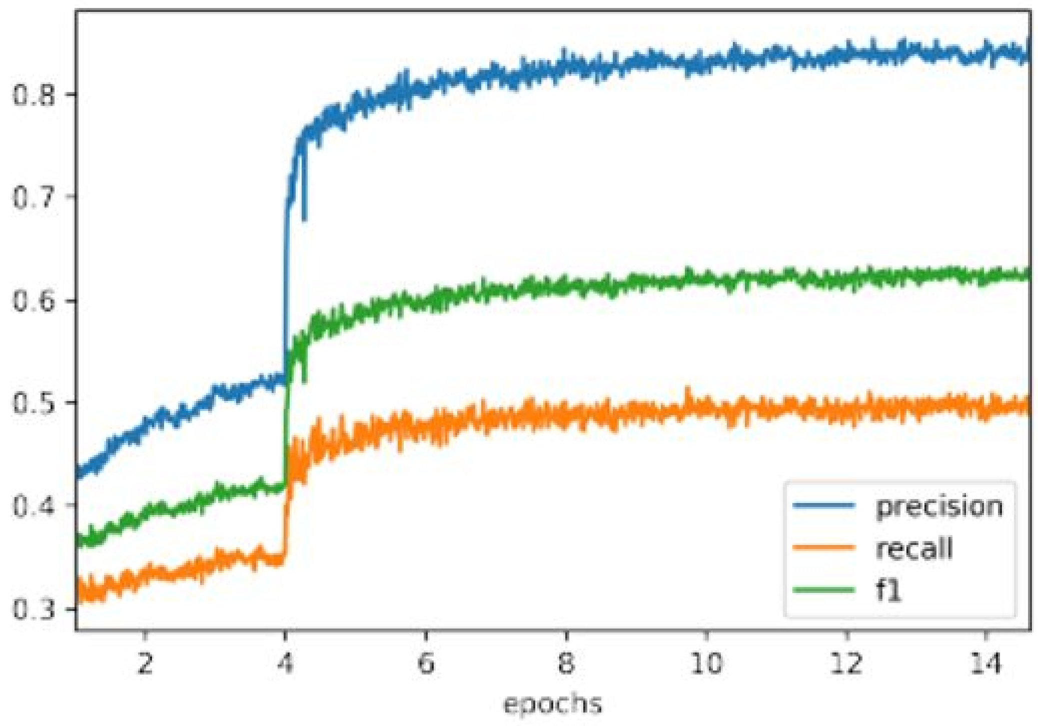
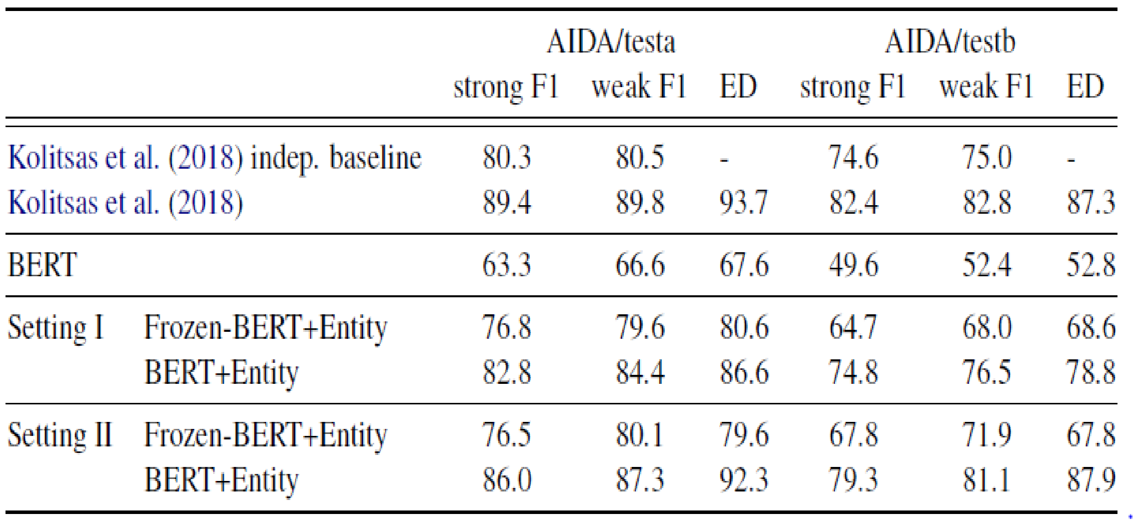


Figure 7: Performance on Setting 2[6]

Figure 7 shows the performance of model and the training progress that is done on validation data in setting 2 for 40 days. As mentioned before, Frozen-BERT+Entity model is trained for first 4 epochs followed by BERT+Entity for remaining epochs out of 14 epochs in setting 2. A steep increase in all the metrics' curves is because of switching the model from Frozen-BERT+Entity to BERT+Entity for training. The performance drastically increases as soon as it is switched to BERT+Entity model. This is a clear evidence of the presence of entity knowledge in an already pre-trained model BERT.

0.6 Results

As mentioned in section 3, Kolitsas et al.is the first to develop a neural model that performs NER and NED jointly. As per its results, the model has achieved the best results among other models that performs the same joint task. Therefore, our results are being compared to Kolitsas et al. that gives the clear picture of how it is performing on entity linking benchmark.



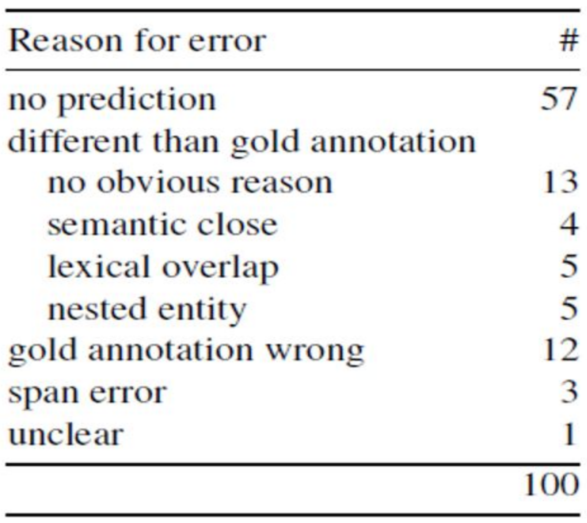
		AIDA/testa			AIDA/testb		
		strong F1	weak F1	ED	strong F1	weak F1	ED
Kolitsas et al. (2018) indep. baseline		80.3	80.5	-	74.6	75.0	-
Kolitsas et al. (2018)		89.4	89.8	93.7	82.4	82.8	87.3
BERT		63.3	66.6	67.6	49.6	52.4	52.8
Setting I	Frozen-BERT+Entity	76.8	79.6	80.6	64.7	68.0	68.6
	BERT+Entity	82.8	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

Figure 8: Comparison of results among all models[6]

CoNLL03/AIDA dataset is divided into two parts namely, testa/AIDA-VALID (AIDA/testa) and testb/AIDA-TEST (AIDA/testb) that are used for validation and testing respectively. From the experimental results (figure 8) it can be observed that significant amount of entity knowledge has been learnt by BERT+Entity when compared to Frozen-BERT+Entity and BERT. Whenever we talk about Frozen-BERT+Entity, it is imperative to know that BERT is not trained and only the entity classifier is trained. Therefore, it is important to know the amount of knowledge the entity classifier has learnt for improving the performance and later the classifier can be tweaked accordingly. This can be computed by the difference of results from BERT and Frozen-BERT+Entity.

Depending on the datasets either Wikipedia or CoNLL03/AIDA, BERT+Entity and Frozen-BERT+Entity shows an increase in scores between $6-10\%$ and $13-16\%$ respectively. The increase in scores that Frozen-BERT+Entity shows is with respect to BERT. The improvement of setting 2 over setting 1 can be observed because the training instances are reduced to relatively low number by removing the constraint of having at least 1 infrequent linked entity and 3 frequent linked entities in setting 1. This allowed setting 2 to have less training instances and ultimately resulted in running more epochs in less training duration.

From the error analysis as given in figure 9, it is found that most of the errors have occurred due to incorrect prediction of entity linking to Nil. This is mostly the case with infrequent entities in the text fragments. Lesser the frequency of occurrence of entities, lesser is the probability of getting linked. Out of 100 errors, 57 errors are due to incorrect prediction of entity links to Nil. Making changes to the model by increasing its layer size could potentially reduce this error.



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

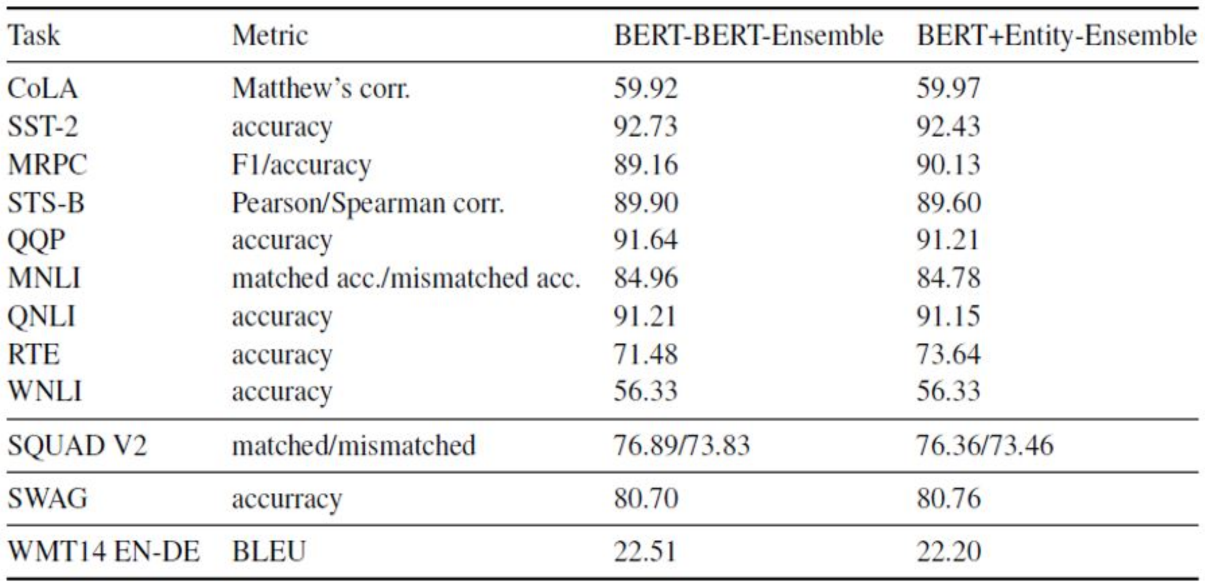
Figure 9: 100 randomly sampled errors from validation data[6]

0.7 Downstream tasks

Downstream tasks are those tasks that make use of a pre-trained model for its accomplishment. Evaluations with respect to setting 1 and setting 2 have been performed on natural language understand task GLUE, the question answering tasks SQUAD V2 and SWAG and the machine translation benchmark EN-DE WMT14 and from the results it has been found that none of the above tasks have performed well or learnt better with BERT+Entity.

Two new models are introduced for this purpose. The outputs of BERT and BERT+Entity are combined to develop BERT+Entity-Ensemble model. Similarly, the results of two BERT models are combined to form BERT-BERT-Ensemble model. Both these models are trained for 3 epochs to evaluate GLUE, SQUAD and SWAG tasks. A special fine-tuned BERT model is used to evaluate the results for machine translation task. The entire BERT model is used as an encoder along with a transformer decoder. This model is known as BERT-2Seq. If BERT+Entity model is used as an encoder with the same transformer decoder then it is known as BERT+Entity-2Seq model. Both these models are trained for 4 epochs in which only encoders are trained in the first epoch followed by training both encoders as well as decoders together in the remaining epochs[6].

The figure 10 shows the results of downstream tasks with BERT+Entity model on setting 1. First 9 subtasks belong to GLUE task followed by SQUAD, SWAG and machine translation tasks. From the experimental results, we can construe that none of the mentioned tasks show an improvement in the score with respect to its previous results trained on different model that resulted in the best score except a subtask of GLUE called RTE task. Arguably we can say that additional entity knowledge is not as beneficial as it should have been for these tasks.



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

Figure 10: Results of downstream tasks in setting 1[6]

0.8 Discussion and Conclusion

0.8.1 Discussion

The main advantage of doing end-to-end entity linking is that it eliminates the problem that is caused by interdependency between NER, candidate generation and NED. For e.g. if NER misses out any entity that should have been recognized then that entity will never be chosen for linking which means that it will never be linked to the knowledge base. The result of BERT+Entity model comes very close to the results of state-of-the-art in end-to-end entity linking[6] i.e. the model developed by Kolitsas et al.. The performance of BERT+Entity shows an increase of 23% – 25% over plain BERT. This shows the impact of BERT+Entity that will have on other applications which use just BERT.

On the other hand, there are few limitations to the model. The performance of all the models drop from AIDA/testa to AIDA/testb due to model overfitting. There are some patterns in the training data that are present in validation data but not in test data and such patterns cause models to result in overfitting[6]. The problem of nil predictions of most of the infrequent entities that is discussed in the previous section is also one of the limitations of this model.

0.8.2 Conclusion

In a nutshell, we can conclude by saying that the performance of BERT+Entity is better than the baselines that model named entity recognition, candidate generation and named entity disambiguation independently one after the other. Among models that perform end-to-end entity linking, BERT+Entity has an edge except Neural Joint Mention Detection and Entity Disambiguation modelled by Kolitsas et al.. It means that this model is the second best to the state-of-the-art in end-to-end entity linking. As the state-of-the-art model performs only NER and NED jointly with a neural approach, BERT+Entity is the first model that learns all the three steps of entity linking jointly with a neural approach. Enhancement of hardware capacity and configuration can have an impact on the performance of BERT+Entity model. Therefore, as a future work, the performance of the model can be evaluated precisely by increasing the hardware. We also came into conclusion that additional entity knowledge will not have any impact on any downstream tasks and it can be opened up for future work as research.

Bibliography

[1] J. Devlin, M. Chang, K. Lee, and K. Toutanova, “BERT: pre-training of deep bidirectional transformers for language understanding,” vol. abs/1810.04805, 2018.

[2] G. Durrett and D. Klein, “A joint model for entity analysis: Coreference, typing, and linking,” *Transactions of the Association for Computational Linguistics*, vol. 2, pp. 477–490, 2014.

[3] N. Kolitsas, O.-E. Ganea, and T. Hofmann, “End-to-end neural entity linking,” in *Proceedings of the 22nd Conference on Computational Natural Language Learning*, (Brussels, Belgium), pp. 519–529, Association for Computational Linguistics, Oct. 2018.

[4] D. B. Nguyen, M. Theobald, and G. Weikum, “J-NERD: Joint named entity recognition and disambiguation with rich linguistic features,” *Transactions of the Association for Computational Linguistics*, vol. 4, pp. 215–229, 2016.

[5] Z. Zhang, X. Han, Z. Liu, X. Jiang, M. Sun, and Q. Liu, “ERNIE: Enhanced language representation with informative entities,” in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, (Florence, Italy), pp. 1441–1451, Association for Computational Linguistics, July 2019.

[6] S. Broscheit, “Investigating entity knowledge in BERT with simple neural end-to-end entity linking,” in *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, (Hong Kong, China), pp. 677–685, Association for Computational Linguistics, Nov. 2019.

[7] A. Sil and A. Yates, “Re-ranking for joint named-entity recognition and linking,” 2013.

[8] I. Yamada, H. Shindo, H. Takeda, and Y. Takefuji, “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*, (Berlin, Germany), pp. 250–259, Association for Computational Linguistics, Aug. 2016.

[9] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, and X. Huang, “Pre-trained models for natural language processing: A survey,” 2020.

[10] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, “Deep contextualized word representations,” in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, (New Orleans, Louisiana), pp. 2227–2237, Association for Computational Linguistics, June 2018.

[11] J. Howard and S. Ruder, “Universal language model fine-tuning for text classification,” in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Melbourne, Australia), pp. 328–339, Association for Computational Linguistics, July 2018.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” *CoRR*, vol. abs/1706.03762, 2017.