CoreLM: Coreference-aware Language Model Fine-Tuning

Nikolaos Stylianou and **Ioannis Vlahavas**
Aristotle University of Thessaloniki
School of Informatics
Thessaloniki, Greece
{nstylia,vlahavas}@csd.auth.gr

Abstract

Language Models are the underpin of all modern Natural Language Processing (NLP) tasks. The introduction of the Transformers architecture has contributed significantly into making Language Modeling very effective across many NLP task, leading to significant advancements in the field. However, Transformers come with a big computational cost, which grows quadratically with respect to the input length. This presents a challenge as to understand long texts requires a lot of context. In this paper, we propose a Fine-Tuning framework, named CoreLM, that extends the architecture of current Pretrained Language Models so that they incorporate explicit entity information. By introducing entity representations, we make available information outside the contextual space of the model, which results in a better Language Model for a fraction of the computational cost. We implement our approach using GPT2 and compare the fine-tuned model to the original. Our proposed model achieves a lower Perplexity in GUMBY and LAMBDADA datasets when compared to GPT2 and a fine-tuned version of GPT2 without any changes. We also compare the models' performance in terms of Accuracy in LAMBADA and Children's Book Test, with and without the use of model-created coreference annotations.

1 Introduction

Language Models (LMs) have seen significant improvements in performance due to the Transformers architecture (Vaswani et al., 2017). The resulting Pretrained Language Models (PLMs) such as BERT (Devlin et al., 2019), GPT2 (Radford et al., 2019) and XLNet (Yang et al., 2019) have contributed to significant advancements in many Natural Language Processing (NLP) tasks. PLMs, regardless of their training objective and methodology, aim to learn contextualized text representations. As such, the available context during training has a key role in the models performance.

The quadratic computation complexity in the attention mechanism of the Transformer architecture, in terms of input sequence length, has been a limiting factor to the amount of contextual information the models can have at each step. To make this architecture more efficient, a plethora of approaches have been introduced (Kitaev et al., 2020; Beltagy et al., 2020; Child et al., 2019, *inter alia*), aiming to lower the computational complexity without sacrificing performance. Tay et al. (2020) summarizes these approaches and categorizes them based on type of attention mechanism in their survey. Yet, even with linear complexity, physical resources will always limit the amount of contextual information that can be handled simultaneously.

In terms of language, entities represent a natural way to tie words together through large pieces of discource. In NLP, Coreference Resolution is the task that aims to identify and group these entity mentions together, when they refer to the same real world entity (Stylianou and Vlahavas, 2021). Therefore, Coreference Resolution presents a natural way to link the context that we are currently handling with distant information, far outside the capabilities model architectures and current hardware resources. However, while this information is present in the text, it is usually not annotated and when annotated are very sparse and in small quantities making it extremely difficult to train large LMs (Kunz and Hardmeier, 2019).

In this paper we present a framework to effectively use coreference annotations to further fine-tune large PLMs, increasing their performance far more than just by fine-tuning on the same data. By using large PLMs we take advantage of existing resources that are expensive to reproduce and come with a big environmental cost (Strubell et al., 2019). What is more, fine-tuning takes advantage of the massive amount of data that essentially initialize the model, making it possible to introduce

Proceedings of the 4th Workshop on Computational Models of Reference, Anaphora and Coreference (CRAC 2021), pages 70–81
Punta Cana, Dominican Republic, November 10–11, 2021.

new capabilities to models with small amounts of annotated data.

In our approach, we use GPT2 as our base model and extend its architecture with the addition of a new Entity-Gating layer that handles entity annotations along with a gating mechanism that handles information flow between the base model and the Entity-Gating layer. As such, our approach uses entity representations when they are available through both training and inference, without imposing any constraints to the model's functionality.

For our experiments, we compare the performance of GPT2, post and pre fine-tuning, with and without our changes in a series of relative tasks. Furthermore, since most coreference annotated datasets are either very small or hard to acquire, we use GUMBY (Gessler et al., 2020) as our fine-tuning dataset which is a model annotated corpus. In addition, by using noisy annotations we aim to show the resilience of our approach to noise and the universality of our framework. Our results highlight the effects of our framework in language modeling, modeling long-range dependencies, and in specific word types where our fine-tuned model achieves better performance than GPT2.

2 Background

This section provides a concise overview of the Transformers architecture (Vaswani et al., 2017), which is the foundation of our approach, followed by a brief explanation of autoregressive language modeling used by our base model, GPT2.

2.1 Transformers

The Transformers architecture is based on stacked Transformer blocks, which take as input a $k \times d$ input vector and return a same size vector after applying a sequence of operations, where k and d denote the context window size and hidden size respectively. Each block is consisted of a multi-head masked self attention layer and a two layer position-wise feed-forward network, each rapped with a layer normalization (`LayerNorm`) layer (Ba et al., 2016) and a residual connection (He et al., 2016). Formally, given an input X the encoder and decoder architecture is described as:

`Encoder:`

$$\begin{aligned} Y &= \texttt{LayerNorm}(\texttt{Self-Attention}(X)) + X \\ Z &= \texttt{LayerNorm}(\texttt{PositionFFN}(Y)) + Y \end{aligned} \quad (1)$$

`Decoder:`

$$\begin{aligned} T &= \texttt{LayerNorm}(\texttt{Self-Attention}(T)) + T \\ P &= \texttt{LayerNorm}(\texttt{Self-Attention}(T, Z)) + T \\ H &= \texttt{LayerNorm}(\texttt{PositionFFN}(P)) + P \end{aligned} \quad (2)$$

However, the decoder can also be used independently by eliminating the second Self-Attention layer.

Self-Attention: The self-attention mechanism takes as input a vector X and projects it into Q, K, V representations for the Query, Key, Value attention scheme. Using the projected vector, this mechanism is formalized as:

$$\texttt{Self-Attention} = \texttt{softmax}(\frac{QK}{\sqrt{d}})V \quad (3)$$

where d is the size of the Q, K, V vectors. Usually the self-attention is multi-headed, in which multiple attentions are calculated in parallel, with the outputs of the multi-headed attentions being concatenated.

PositionFFN: Given an input vector X, this layer applies two position-wise linear transformations with a ReLU activation in between. The PositionFFN layer is formalized as:

$$\texttt{PositionFFN} = max(0, XW_1 + b_1)W_2 + b_2 \quad (4)$$

with W_1, b_1 and W_2, b_2 being the trainable weight and bias parameters of each layer respectively.

2.2 Autoregressive Language Modeling

Autoregressive language models estimate the distribution over a sequence of word tokens by factorizing their joint probabilities as the product of conditional probabilities (Bengio et al., 2003). For a context vector of tokens $U = (u_1, \ldots, u_k)$, this is formally described as:

$$p(U) = \prod_{i=1}^{k} p(u_k | u_1, \ldots, u_{k-1}) \quad (5)$$

where, k is the context window size.

GPT2 is an autoregressive LM, based on the previously described Transformer's decoder architecture. In comparison to the previously described architecture, it uses masked multi-headed attention to prevent leftward information flow in the decoder. For further information about the model's architecture we refer readers to Radford et al. (2019).

3 Coreference-aware Language Modeling

Language models are limited to the amount of information they can process based on the available context window k (eq. (5)). However, increasing k will lead to an exponential increase in computation complexity. As such, we implicitly increase the available information, without increase the context window, using coreference annotations.

We do this by introducing entity-representations (§3.2), in the form of vectors, that are utilized by the model to infuse the respective entity information to all the tokens in the sequence that are part of entities. The entity-representations are created from the whole discourse available, hence holding contextual information that are very distant to the context window. As such, we make no alterations to the language modeling objective. These vector representations are introduced to the model via an Entity-Gating layer (§3.3) that is added to model architecture.

3.1 Architecture

Our base model, GPT2, is comprised of N stacked Transformer decoder blocks, where each is consisted of a multi-headed attention layer and a position-wise feedforward layer with residual connections and layer normalizations. We extend its architecture by adding an Entity-Gating layer after the Transformer decoders (eqs. (6) to (9)).

$$h_0 = UW_e + W_p \tag{6}$$

$$h_l = \texttt{transformer\_decoder}(h_{l-1}) \forall_i \in [1, n] \tag{7}$$

$$h_e = \texttt{entity\_gating}(h_n, E) \tag{8}$$

$$p(U) = \texttt{softmax}(h_e W_e^T) \tag{9}$$

Here, n is the number of layers, W_e is the token embedding matrix, W_p is the position embedding matrix and E is the context vector of entity representations. Figure 1 illustrates a high level view of the model of our proposed architecture.

3.2 Entity representations

Each entity is represented by a learned vector $E_i \in \mathbb{R}^{1 \times d_{embd}}$, where d_{embd} is the embedding dimension of the model (W_e). These entity vectors are stored in a persistent set of entities $\mathcal{E}$ so that they can be utilized through out the whole discourse scope. We use E_0 as a static entity representation for tokens that are not part of an entity.

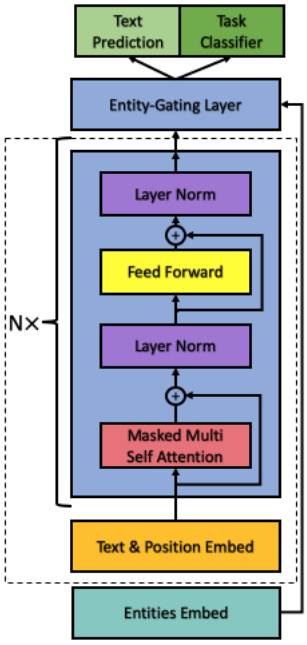

Figure 1: CoreLM Architecture as an extention of GPT2 model (area within dashed border).

The entity representations are initialized as a vector of ones. This design choice is based on the architecture of the Entity-Gating layer (§3.3) and the learning process. Specifically, as each token is accompanied with an entity representation. Initializing the entity representations this way introduces less noise during the first occurrence of an entity mention. It also provides a dynamic way of using the same architecture, even when no entity are available.

3.3 Entity-Gating

Our proposed Entity-Gating layer (Figure 2) follows the same design principles of the GPT2 Transformer decoder blocks, using a Multi-Headed Attention layer, Layer Normalization layers and residual connections. However, we replace the Masked Multi-Head Self Attention layer with an Entity-Attention layer and use a learnable gating mechanism to control flow of information. Formally, the Entity-Gating layer is described as:

$$EG_A = \texttt{LayerNorm}(\texttt{EntityAttention}(h_l)) + h_l \tag{10}$$

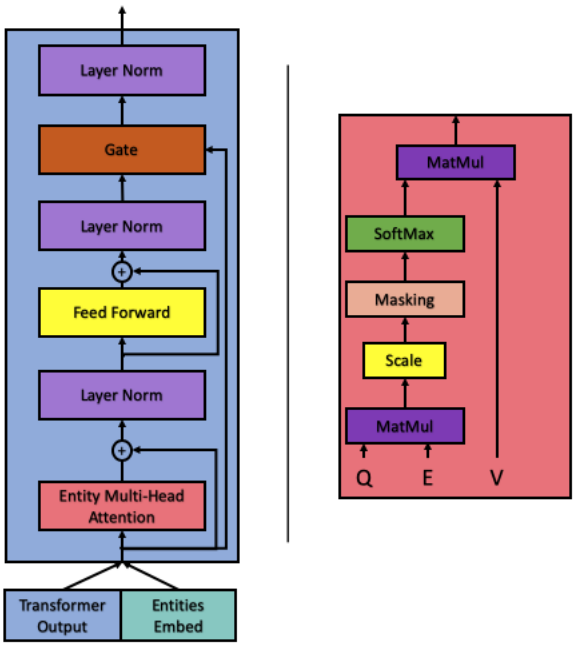

Figure 2: **(left)** Entity-Gating layer architecture. **(right)** Entity-Attention mechanism. In the illustration, we assume a single attention head for simplicity.

$$EG_B = \texttt{LayerNorm}(\texttt{PositionFFN}(EG_A)) + EG_A \tag{11}$$

$$h_e = \texttt{LayerNorm}(\texttt{Gate}(EG_B, h_l)) \tag{12}$$

The `LayerNorm` and `PositionFFN` are used as described in Section 2.1, from the original architecture.

Our `EntityAttention` layer uses the Query (Q), Entity (E), Value (V) scheme described in eq. (13) (Stylianou and Vlahavas, 2020). In comparison to their variation, we use a multi-headed approach so that we limit the effect of entity representation to the tokens closer to the entity mention. As such, the attention mechanism is defined as:

$$EntityAttention = \texttt{Softmax}(\frac{QE}{\sqrt{d_k}})V \tag{13}$$

where d_k is the dimension of the queries and entities in each head. Finally, the `Gate` layer combines the layer input h_l with EG_B using the following gating mechanism before applying a layer normalization to the output:

$$g_e = \delta(\sigma(Vh_l)) \tag{14}$$

$$z_e = (1 - g_e) \odot EG_B + g_e \odot h_l \tag{15}$$

in which σ is a sigmoid function, V is a parameter vector, $\odot$ is the Hadamard product and δ is a gate flow rate. We use a gate flow rate to ensure that the entity information is considered by our final model. By definition, $\delta \to 0$ results in no pass-through of information from the gate, $\delta \to 1$ results in completely dependent pass-through from the learned vector and $\delta \to 0.5$ enforces at least a fifty-fifty split of information.

4 Experiments

We Fine-Tune our entity-aware LM with the original training objective of maximizing the log-probability of U (eq. (5)). Language Models are evaluated in terms of Perplexity (PPL) which is the exponentiated average negative log probability per word prediction, as we are using a word-level base model. As such, PPL is a direct reflection of the model's loss. For Fine-Tuning we use GUMBY, a model annotated coreference corpus, containing 4960 documents.

The entity-aware fine-tuned LM is evaluated on GUMBY, using the created entity representations during Fine-Tuning and in a Zero-Shot setting, without any further training, on LAMBADA, WikiText2, WikiText103 in terms of PPL and on LAMBADA and CBT in terms of Accuracy. We also evaluate the effect of newly introduced entity annotations from a separate Coreference Resolution model on the Zero-Shot evaluated corpora, to investigate their effect in the model's performance. Detail information about the model parameters and experimental setup are provided in Appendix A. Appendix B enlists our methodology to annotate the LAMBADA and CBT corpora with coreference clusters, used in Zero-Shot evaluation only.

4.1 Fine-Tuning

In order to fine-tune the model, using the annotated entity information in the GUMBY corpus (Gessler et al., 2020), we re-formatted it by introducing a second input stream along the raw text, which assigns a unique entity identifier in the corresponding token in the text (Stylianou and Vlahavas, 2020).

We similarly use $\varnothing$ to identify tokens that are not part of an entity. However, in order to utilize the encapsulated entity information present, we create multiple instances of the source files each annotated with a single layer of entity annotations. This comes in comparison with their approach in which only the outer entities are considered. We

Table 1: Language Modeling performance evaluation on fine-tuning and zero-shot setting. The fine-tuned models are trained only on GUMBY and evaluated on LAMBADA, WikiText2 and WikiText103 test sets.

	GUMBY (PPL)	LAMBADA (PPL)	WikiText2 (PPL)	WikiText103 (PPL)
GPT2 Zero-Shot	66.19	26.49	**30.63**	35.81
GPT2 +Fine-Tuning	52.60	28.93	32.36	35.53
GPT2 +CoreLM +Fine-Tuning	**46.97**	**26.21**	31.80	**29.51**

treat all entities as encapsulated entities, so that the second instance of the source file is only annotated with the entities that were identified within the span of text of first entity layer (example in Appendix A - Table 3). As a result, our final data was consisted of 13070 documents, which is approximately triple the original size. As the original corpus does not provide with predefined split, we held out 10% of the data for evaluation while maintaining the balance between source types.

The fine-tuned model is compared against the base model (GPT2) and a fine-tuned version of the base model without any entity information, on the held out data of GUMBY. As shown in Table 1 our approach significantly improves the model's performance after fine-tuning compared to the original model.

4.2 Language Modeling

For the Language Modeling evaluation we use WikiText2, WikiText103 (Merity et al., 2016) and LAMBADA (Paperno et al., 2016) in a Zero-Shot evaluation setting. We compare the previously fine-tuned entity-aware LM with the base-model, with and without any Fine-Tuning on GUMBY. The results are showcased in Table 1.

We notice that fine-tuning GPT2 with GUMBY does not generalize well in other domains, leading in a significant drop in performance in both LAMBADA and WikiText2. In WikiText103, in which the unknown words are proper names, the fine-tuned version performs better. With CoreLM, our model avoids catastrophic forgetting and presents similar performance gains in all corpora, resulting in a slight improvement in LAMBADA and WikiText103 and a slight impairment in WikiText2 compared to the base model.

4.3 LAMBADA

The LAMBADA corpus is used to evaluate our approach, in a Zero-shot setting. LAMBADA is designed to test the model's ability to use long range dependencies in text. Long range dependency, in this context, is consider to be a context window of 50 tokens, which the architecture can handle due to the 1024 token window context. During evaluation, we use use model acquired coreference annotation to the corpus, using a pretrained model Coreference Resolution model (Appendix B). Hence, we also evaluate the effect of coreferent information.

Without Coreference: Comparing the model's performance to GPT2, our approach achieves increased Accuracy in correctly predicting the last word, with scores 46.67% for GPT2 and 48.11% for CoreLM (statistically significant with paired t-test $p << 0.01$ - Table 2). This increase in performance is slightly bigger if we compare it to the fine-tuned GPT2 model on GUMBY.

With Coreference: Coreference annotations offer a slight increase in performance, with 0.28% Accuracy increase compared to the CoreLM version without coreference annotations. While this increase is not a statistically significant contribution, it is an expected behavior given that the entity representations were initialized during the zero-shot evalutation (Appendix B). Furthermore, the context of each of LAMBADA entries is well inside the capabilities of the GPT2 model architecture and consequently the coreference annotations did not provide any information that were not already accessible by the model.

4.4 Children's Book Test

The Children's Book Test (CBT) (Hill et al., 2016) was designed to evaluate LMs in different word cat-

Table 2: Zero-Shot evaluation on LAMBADA and Children's Book Test (CBT) with and without coreference annotations.

	LAMBADA (Acc)	CBT-CN (Acc)	CBT-NE (Acc)	CBT-V (Acc)	CBT-P (Acc)
GPT2 Zero-Shot	46.67	**84.48**	64.52	92.24	91.00
GPT2 +Fine-Tuning	46.63	84.02	64.24	92.04	90.80
GPT2 +CoreLM +Fine-Tuning	48.11	84.16	64.48	**92.40**	90.88
GPT2 +CoreLM +Fine-Tuning +Coref	**48.39**	84.16	**64.56**	**92.40**	**91.97**

egories. In comparison to GPT2, we report scores in all categories, i.e. Common Nouns (CN), Named Entities (NE), Verbs (V) and Prepositions (P). CBT is designed as a cloze test, in which a hidden word should be predicted, given ten possible options. We formulate this task as a language modeling task, similar to Radford et al. (2019), in which we condition the sentence with each option and calculate its probability, choosing the one with the highest probability as the final prediction. Similarly to LAMBADA, we use a pretrained Coreference Resolution model to annotate the corpus with newly initialized entity mentions.

Table 2 shows the results in terms of Accuracy with and without the use of coreference annotations.

Without Coreference: Comparing the base model, with a fine-tuned version of the base model and CoreLM, it becomes obvious that the GUMBY does not generalize well with CBT. As a result, we note a drop in performance by just fine-tuning the model in all word categories. However, the CoreLM fine-tuned version results in better performance compared to the GPT2 fine-tuned model, with insignificant differences from the base model in almost all categories, with the exceptions of Verbs in which we notice a slightly better Accuracy.

With Coreference: Including coreference annotation to the CoreLM model results in small changes in performance in all categories. Specifically, there is no gain in performance when using entities in CN corpus variant. However, in the NE variant we achieve a 0.16% increase. This very small increase is because in the majority of the cases, the cloze test answers are similar to the surface forms of the entities as found in the context and as such the correlation is very easy for model to make without the need of extra information. For the V variant of the corpus, there is no change to the performance while using coreference annotations as expected, while for the P variant, there is a increase of 1.09% (statistically significant with t-test $p < 0.01$). This increase in prepositions is attributed to the accurate resolution of the nouns as mentions of entities that changed their representations.

The performance changes, while using coreference annotation are indicative of the impact of entity representations in CoreLM. As our base model is capable of contextualizing each entry in CBT due to it's context window, we expect these improvements to be bigger in corpora which the information outside the context window of the model is required.

5 Error Analysis

In our experiments, we showcased the effects of CoreLM on the base model, in different scenarios. In this section we investigate the cases in which the base model performed better than the CoreLM model and vice-versa. For that reason, we manually compared the predictions made between the two models, using the GUMBY Fine-Tuned GPT2 model as the deciding factor in our observations between the effects of CoreLM Fine-Tuning and simple Fine-Tuning. We limit our analysis on the

CBT corpus which provides a meaningful way to evaluate the changes due to its word category variants.

In our analysis, we noticed that fine-tuning on GUMBY lead to wrong choices by the model in all categories, which were not made by the GPT2. However, when CoreLM was used with coreference annotations, 83% of those cases were corrected. The vast majority of the corrected cases were in the Prepositions and Named Entities word categories, with only 19% of the corrected ones in Verbs and Common Noun word categories. A positive correlation between corrected cases and correct coreference annotations was noticed, as cases in which CoreLM persisted on the wrong option also had different coreferent annotations than the correct sentence.

In the cases where CoreLM performed better than both the fine-tuned and base model, we noticed the same correlation between coreference annotations that directly affected the available options. Unavailingly, a small number of cases in which, the nouns following a preposition or the Named Entities themselves were annotated in a wrong coreference cluster, lead to different probability distributions for the available options and eventually to making the wrong selection. Furthermore, comparing the predictions made by CoreLM with and without the use of coreference annotations, we notice that the majority of the errors persisted when the option was not directly annotated in a coreference cluster.

6 Discussion

Our approach takes advantage of PLMs and increases their performance by exploiting distant contextual information in the form of entity representations using coreference annotations. What is more, it can be used with and without the existence of such information, hence not hindering the flexibility of PLMs. Our experiements when Fine-Tuning GPT2 and the GPT2 with CoreLM on the same data, without using coreference information show that even when the fine-tuning data are not best suited for the downstream tasks, CoreLM maintains more of the original model's performance, making it a more resilient Fine-Tuning methodology. In addition, as CoreLM is very modular, it can be applied to the majority of LMs, including non-autoregressive approaches such as BERT.

In Language Modeling, our approach achieved significantly lower Perplexity in all corpora compared to the GPT2 Fine-Tuned version. What is more, GUMBY proved to be an ill-suited corpus to fine-tune for both LAMBADA and WikiText based on the post Fine-Tuning performance. Regardless, Fine-Tuning with CoreLM demonstrated significant gains, even compared to the pre Fine-Tuned model.

In both LAMBADA and CBT, we show increase in Accuracy compared to GPT2 pre and post Fine-Tuning. Most notably, the Named Entity and Prepositions word types showed the biggest increase in CBT, with Common Nouns suffering in comparison compared to the pre Fine-Tuned version and Verbs attaining a slightly better Accuracy. Our error analysis highlights the effect of coreference annotations to these changes in performance. In all cases, Fine-Tuned GPT2 achieved lower scores in all word categories compared to CoreLM Fine-Tuned, which indicated that GUMBY is not a suitable fine-tuning corpus for these tasks.

Using model-created coreference annotations during Zero-Shot evaluation did increase the performance slightly. While the performance increase is minor, the entity representations used were only initialized from the scope of each example as there was not long contextual information to take advantage off. What is more, coreference annotations increased the performance regardless of the information being within the context window in all the examples, indicating that further gains can be achieved by using coreference annotations extensively over large pieces of discourse.

Unavailingly, our approach is based on the ability of other models to accurately predict coreference clusters so that CoreLM can exploit the coreferent mentions. The errors in the predicted clusters introduce noise, to both entity representations and the final model. What is more, maintaining a persistent set of entity representations, is computationally expensive and can be very burdening when considering a large collection of documents. As a result, an entity management mechanism, similar to the one used in Toshniwal et al. (2020), would be required for CoreLM to scale efficiently to bigger document collections.

7 Related Work

Early entity-aware LMs were trained from scratch with entity information available through the training process. Specifically, Yang et al. (2017) and Ji

et al. (2017) both introduced models that used reference information with attention-based mechanisms to incorporate them into the model. Yang et al.'s (2017) model made use of both intra-linguistic, coreferring mentions in text, and extra-linguistic, tables and lists, features through three different components, only creating learnable embeddings for intra-linguistic features. Ji et al.'s (2017) model was focused only on intra-linguistic mentions, i.e. corefererring mentions, and introduced additional control variables indicating if the next token is part of an entity as well as the number of remaining entity tokens. Recently, Kunz and Hardmeier (2019) extended Yang et al.'s (2017) approach by using learnable entity embeddings. These approaches also have the ability to autoregressively predict the entity of the following word and constrain the word generation to a specific entity.

EnGen combines EntityNLM (Ji et al., 2017) with S2SA (Sutskever et al., 2014), to train a generative language model that uses both previous sentence representations and entity representations in order to generate coherent text (Clark et al., 2018). In comparison to their approach, our approach handles entities and information flow differently. Stylianou and Vlahavas (2020) also introduced a Transformer-based approach towards incorporating learnable entity representations, using multi-head attentions inside all the Transformer blocks of the model. In comparison to past approaches, this approach was only focused on the effective use of entity information in a LM and did not predict the following entity information. However, all of these models required high quality annotated data to be trained from scratch which were limited and of specific genre (Kunz and Hardmeier, 2019; Stylianou and Vlahavas, 2020).

Other methods have focused on using only extra-linguistic information, ignoring pronouns and nominal mentions in text. ERNIE (Zhang et al., 2019) uses Knowledge Graphs (KG) to extract entities for identified named entities. However, ERNIE represents entity information using a pre-trained knowledge embedding model, trained on the used KG, and does not create dynamic entity representations. Liu et al. (2019) use Knowledge Bases to learn word type embeddings based on the learned type representations. Similar to past approaches (Ji et al., 2017; Kunz and Hardmeier, 2019) it can autoregressively constrain the prediction to a certain entity type. The type information is restricted using pre-defined vocabularies making the model less dynamic in its entity predictions. Both approaches have been found to be very effective for the tasks that were respectively designed, however they lack in expandability of their domain of application without requiring complete retraining.

8 Conclusions and Future Work

In this paper we presented CoreLM, a modular Fine-Tuning framework for PLMs to exploit model-created coreference annotations in order to create better mention representations and an overall better LM. In our experiments we showcased a performance increase when evaluating in a zero-shot setting, compared to the similarly fine-tuned model, even when the fine-tuning corpus did not generalize well to the end tasks. Our analysis shows that coreference annotations play a significant role in both Fine-Tuning and in downstream task performance, with correct annotations leading to better performance when used.

In addition, our work helps in adding a new frontier to Coreference Resolution through the effective use of coreference annotations in Language Modeling. In this paper we showcased the effects of coreference annotation even when the information is within the context window of the model. Using coreference annotations can further lead to the decrease of the required context window and boost approaches like Shortformer (Press et al., 2020), leading to better and more efficient LMs.

In the future we aim to create a more efficient approach to LM through the use of both intra-linguistic (Coreference) and extra-linguistic (KG) features. Undeniably, KGs provide a means for structured, high quality information that cannot be found in a single text. We believe that an information fusion from coreference annotation and graph nodes, along with short context window will not be computationally prohibitive and lead in better, information rich, LMs.

Acknowledgements

This research is co-financed by Greece and the European Union (European Social Fund - ESF) through the Operational Programme "Human Resources Development, Education and Lifelong Learning" in the context of the project "Strengthening Human Resources Research Potential via Doctorate Research" (MIS-5000432), implemented by the State Scholarships Foundation (IKY).

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv:2004.05150*.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *The journal of machine learning research*, 3:1137–1155.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

Elizabeth Clark, Yangfeng Ji, and Noah A. Smith. 2018. Neural text generation in stories using entity representations as context. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2250–2260, New Orleans, Louisiana. Association for Computational Linguistics.

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, Minneapolis, Minnesota. Association for Computational Linguistics.

Luke Gessler, Siyao Peng, Yang Liu, Yilun Zhu, Shabnam Behzad, and Amir Zeldes. 2020. Gumby–a free, balanced, and rich english web corpus. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 5267–5275.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The goldilocks principle: Reading children's books with explicit memory representations. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

Yangfeng Ji, Chenhao Tan, Sebastian Martschat, Yejin Choi, and Noah A. Smith. 2017. Dynamic entity representations in neural language models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1830–1839, Copenhagen, Denmark. Association for Computational Linguistics.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*.

Jenny Kunz and Christian Hardmeier. 2019. Entity decisions in neural language modelling: Approaches and problems. In *Proceedings of the Second Workshop on Computational Models of Reference, Anaphora and Coreference*, pages 15–19, Minneapolis, USA. Association for Computational Linguistics.

Angli Liu, Jingfei Du, and Veselin Stoyanov. 2019. Knowledge-augmented language model and its application to unsupervised named-entity recognition. 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 1142–1150, Minneapolis, Minnesota. Association for Computational Linguistics.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. 2020. On the variance of the adaptive learning rate and beyond. In *International Conference on Learning Representations*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1525–1534, Berlin, Germany. Association for Computational Linguistics.

Ofir Press, Noah A Smith, and Mike Lewis. 2020. Shortformer: Better language modeling using shorter inputs. *arXiv preprint arXiv:2012.15832*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. Zero: Memory optimizations toward training trillion parameter models. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '20. IEEE Press.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, page 3505–3506, New York, NY, USA. Association for Computing Machinery.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.

Nikolaos Stylianou and Ioannis Vlahavas. 2020. E.T.: Entity-transformers. coreference augmented neural language model for richer mention representations via entity-transformer blocks. In *Proceedings of the Third Workshop on Computational Models of Reference, Anaphora and Coreference*, pages 1–10, Barcelona, Spain (online). Association for Computational Linguistics.

Nikolaos Stylianou and Ioannis Vlahavas. 2021. A neural entity coreference resolution review. *Expert Systems with Applications*, 168:114466.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2020. Efficient transformers: A survey. *arXiv preprint arXiv:2009.06732*.

Shubham Toshniwal, Sam Wiseman, Allyson Ettinger, Karen Livescu, and Kevin Gimpel. 2020. Learning to Ignore: Long Document Coreference with Bounded Memory Neural Networks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8519–8526, Online. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Zichao Yang, Phil Blunsom, Chris Dyer, and Wang Ling. 2017. Reference-aware language models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1850–1859, Copenhagen, Denmark. Association for Computational Linguistics.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451, Florence, Italy. Association for Computational Linguistics.

Table 3: Data example from the GUMBY corpus, as formatted for the task. Other corpora are similarly formatted, with multiple rows of entity annotation when coreference information were needed.

$X_{1:12}$	The	prime	minister	of	Israel	,	Binyamin	Netanyahu	,	told	a	news
$E_{1:12}$	11	11	11	11	11	∅	11	11	∅	∅	13	13
$E_{1:12}$	∅	∅	∅	∅	7	∅	∅	∅	∅	∅	∅	∅

A Experimental Setup

In all our experiments we use the GPT2-small configuration with 124M parameters, with 12 layers and 12 attention heads each for our base model. We add one Entity-Gating layer after the base model's Transformer layers, which has a masked multi-headed entity attention layer with 12 heads and a 10% dropout between layers. The gate flow rate (δ) is set to 0.5. These hyperparameters were found to perform best from [6, 8, 10, 12, 16] number of heads and [0.2, 0.5, 0.7, 1] gate flow rate (δ) after manual tuning.

All datasets are tokenized using the pre-trained GPT2 tokenizer, which uses Byte Pair Encoding (BPE) (Sennrich et al., 2016). We also apply a simple de-tokenization based on author's responses in the official GitHub repository as the exact de-tokenizer used to achieve the published results has not been made available.[1] We use the OpenAI LAMBADA split for evaluation and remove the The Jungle Book by Rudyard Kipling from CBT as it was found to be part of the GPT2 original training set (Radford et al., 2019). As such, all scores are based on our own experiments and in some cases vary (both positively and negatively) from the reported scores.

Our model has 132M parameters, a 6% increase, after the addition of the Entity-Gating layer and the entity representations. It is fine-tuned on the GUMBY corpus for 10 epochs, with a batch size of 128. Rectified Adam (Liu et al., 2020) was used as the optimizer with 100 steps of warm up and a linearly decaying learning rate with a starting value of 1e-5. During Fine-Tuning, the entity representations are updated after every training step. We freeze all 12 GPT2 Transformer layers and apply gradients only to the input layers (W_e and W_p), output layers (the language modeling head) and the Entity-Gating layer. During Zero-Shot evaluation we do not use any entity information and as such we discard the persistent entity representations.

All experiments were run on a single Titan V 12GB graphics card, using half precision floating-point format, Zero Stage 2 optimization (Rajbhandari et al., 2020) and DeepSpeed (Rasley et al., 2020). In this setup, fine-tuning takes approximately 8 hours, with no noticeable differences in terms of inference speed compared to GPT2.

[1] https://github.com/openai/gpt-2/issues/131

B Coreference Annotations

The vast majority of the datasets do not come with coreference annotations, a process which is very expensive and time consuming if it was to be done by human annotators. The same issue rises from the use of free text from web sources. In order to fully exploit our proposed framework, we uses the pre-trained Coreference Resolution model by Toshniwal et al. (2020) to create noisy coreference annotations for LAMBADA and Children's Book Test (CBT) corpora.

Our approach does not have an entity-linking component with which the originally identified entities from the GUMBY corpus could be linked with newly identified entities in the other corpora. As such, the persistent entity representation used in the original GUMBY corpus were reset for each corpora. Hence, the resulting entity representation are not as descriptive as the GUMBY created ones as, there was little context involved and we only used the corpora for zero-shot evaluation, not allowing for iteratively creating richer entity representations.

LAMBADA: No major preprocessing was required for the LAMBADA dataset. We did treat each entry in the dataset as a new document so that coreference annotations did not point to entities on other unrelated entities. The resulted entity representation were created as an average of the hidden representation of all the entity mention in that entry (§3.2), excluding the last word and its entity annotation which were to be predicted.

Children's Book Test: For CBT, we first formulated the corpus to fit our Language Modeling approach (§4.4), conditioning each choice with one of 10 possible candidates and annotating the document as if the candidate was the answer in the cloze test. During evaluation, we predicted coref-

erence clusters for the context conditioned with all possible candidate choices.