COM SCI 263:
DeBERTa: Decoding-enhanced BERT with Disentangled Attention [2]

Reading Notes

Tejas Kamtam
tejaskamtam@ucla.edu

Summary

Overview and Related Works

With the rise of pre-trained LLMs, the race to make the *best* language model incentivized novel approaches to modeling attention in transformers. DeBERTa (**D**ecoding-**e**nhanced BERT with disentangled **a**ttention) [2] tackles this challenge by separating the position and word embedding as inputs to the model and generating disentangled attention matrices which are collectively summed to generate contextualized vectors. DeBERTa is an encoder-only model trained on the Masked Language Modeling (MLM) task to improve on baselines set by BERT [1] and RoBERTa [3]. Secondly, DeBERTa takes a novel approach to position consideration in MLM by considering absolute position of tokens to distinguish context for inputs with ambiguity in relative position. Finally, the paper expands beyond training on natural language understanding (NLU) and natural language generation (NLG) tasks, to also provide a robust framework for adversarial training and preventing adversarial perturbation attacks.

Disentangled Attention

As a reminder, the standard self-attention formulation [5] is given as:

$$\boldsymbol{Q} = \boldsymbol{H}\boldsymbol{W_q}, \boldsymbol{K} = \boldsymbol{H}\boldsymbol{W_k}, \boldsymbol{V} = \boldsymbol{H}\boldsymbol{W_v}, \boldsymbol{A} = \frac{\boldsymbol{Q}\boldsymbol{K}^\intercal}{\sqrt{d}}$$

$$\boldsymbol{H_o} = \text{softmax}(\boldsymbol{A})\boldsymbol{V}$$

"where $\boldsymbol{H} \in \mathbb{R}^{N\times d}$ represents the input hidden vectors, $\boldsymbol{H_o} \in \mathbb{R}^{N\times d}$ the output of self-attention, $\boldsymbol{W_q}, \boldsymbol{W_k}, \boldsymbol{W_v} \in \mathbb{R}^{d\times d}$ the projection matrices, $\boldsymbol{A} \in \mathbb{R}^{N\times N}$ the attention matrix, N the length of the input sequence, and d the dimension of hidden states."

DeBERTa's attention formulation considers a token at position i in a sequence, "using two vectors, that can be represented as $\{\boldsymbol{H_i}\}$ and $\{\boldsymbol{P_{i|j}}\}$ which represent its content and relative position with the token at position j, respectively." Then, the attention matrices can be found as 4 unique components:

$$\begin{aligned} A_{i,j} &= \{\boldsymbol{H_i}, \boldsymbol{P_{i|j}}\} \times \{\boldsymbol{H_j}, \boldsymbol{P_{j|i}}\}^\intercal \\ &= \boldsymbol{H_i}\boldsymbol{H_j}^\intercal + \boldsymbol{H_i}\boldsymbol{P_{j|i}}^\intercal + \boldsymbol{P_{i|j}}\boldsymbol{H_j}^\intercal + \boldsymbol{P_{i|j}}\boldsymbol{P_{j|i}}^\intercal \end{aligned}$$

"That is, the attention weight of a word pair can be computed as a sum of four attention scores using disentangled matrices on their contents and positions as *content-to-content*, *content-to-position*, *position-to-content*, and *position-to-position*"
Then the disentangled self-attention can be represented with relative position bias as (1), "where $\boldsymbol{Q_c}, \boldsymbol{K_c}$ and $\boldsymbol{V_c}$ are the projected content vectors generated using projection matrices $\boldsymbol{W_{q,c}}, \boldsymbol{W_{k,c}}, \boldsymbol{W_{v,c}} \in R^{d\times d}$ respectively, $\boldsymbol{P} \in R^{2k\times d}$ represents the relative position embedding vectors shared across all layers (i.e., staying fixed during forward propagation), and $\boldsymbol{Q_r}$ and $\boldsymbol{K_r}$ are projected relative position vectors generated using projection matrices $\boldsymbol{W_{q,r}}, \boldsymbol{W_{k,r}} \in R^{d\times d}$, respectively." Using these projected matrices, calculating the contextualized vector representations mirror that of the

standard formulation:

$$\begin{gathered}\boldsymbol{Q_c} = \boldsymbol{HW_{q,c}}, \boldsymbol{K_c} = \boldsymbol{HW_{k,c}}, \boldsymbol{V_c} = \boldsymbol{HW_{v,c}}, \boldsymbol{Q_r} = \boldsymbol{PW_{q,r}}, \boldsymbol{K_r} = \boldsymbol{PW_{k,r}} \\ \tilde{A}_{i,j} = \underbrace{\boldsymbol{Q_i^c}\boldsymbol{K_j^c}^\intercal}_{\text{(a) content-to-content}} + \underbrace{\boldsymbol{Q_i^c}\boldsymbol{K}^r_{\boldsymbol{\delta(i,j)}}{}^\intercal}_{\text{(b) content-to-position}} + \underbrace{\boldsymbol{K_j^c}\boldsymbol{Q}^r_{\boldsymbol{\delta(j,i)}}{}^\intercal}_{\text{(c) position-to-content}} \\ \boldsymbol{H_o} = \text{softmax}(\frac{\tilde{\boldsymbol{A}}}{\sqrt{3d}})\boldsymbol{V_c}\end{gathered} \tag{1}$$

"where $\tilde{A}_{i,j}$ is the element of attention matrix $\tilde{\boldsymbol{A}}$, representing the attention score from token i to token j. $\boldsymbol{Q_i^c}$ is the i-th row of $\boldsymbol{Q_c}$. $\boldsymbol{K_j^c}$ is the j-th row of $\boldsymbol{K_c}$. $\boldsymbol{K}^r_{\boldsymbol{\delta(i,j)}}$ is the $\delta(i,j)$-th row of $\boldsymbol{K_r}$ with regarding to relative distance $\delta(i,j)$. $\boldsymbol{Q}^r_{\boldsymbol{\delta(j,i)}}$ is the $\delta(j,i)$-th row of $\boldsymbol{Q_r}$ with regarding to relative distance $\delta(j,i)$." Where, "[denoting] k as the maximum relative distance, $\delta(i,j) \in [0, 2k)$ is the relative distance from token i to token j, which is defined as:"

$$\delta(i,j) = \begin{cases} 0 & \text{for} \quad i-j \leq -k \\ 2k-1 & \text{for} \quad i-j \geq k \\ i-j+k & \text{others.} \end{cases}$$

Efficiency

Standard computation of disentangled attention as given above requires $O(N^2 d)$ space where N is the sequence length and d is the hidden dimension size of the inputs. The authors of the paper have created an efficient algorithm below to reduce the space requirement to $O(kd)$ where k is the maximum possible relative distance (context window) which is capped at 512 for DeBERTa small, base, and large models.

Algorithm 1 Disentangled Attention

Input: Hidden state $\boldsymbol{H}$, relative distance embedding $\boldsymbol{P}$, relative distance matrix $\boldsymbol{\delta}$. Content projection matrix $\boldsymbol{W_{k,c}}$, $\boldsymbol{W_{q,c}}$, $\boldsymbol{W_{v,c}}$, position projection matrix $\boldsymbol{W_{k,r}}$, $\boldsymbol{W_{q,r}}$.

$\boldsymbol{K_c} = \boldsymbol{HW_{k,c}}, \boldsymbol{Q_c} = \boldsymbol{HW_{q,c}}, \boldsymbol{V_c} = \boldsymbol{HW_{v,c}}, \boldsymbol{K_r} = \boldsymbol{PW_{k,r}}, \boldsymbol{Q_r} = \boldsymbol{PW_{q,r}}$
$\boldsymbol{A_{c\to c}} = \boldsymbol{Q_c}\boldsymbol{K_c}^\intercal$
for $i = 0, ..., N-1$ **do**
    $\tilde{\boldsymbol{A}}_{\boldsymbol{c\to p}}[i,:] = \boldsymbol{Q_c}[i,:]\boldsymbol{K_r}^\intercal$
end for
for $i = 0, ..., N-1$ **do**
    for $j = 0, ..., N-1$ **do**
        $\boldsymbol{A_{c\to p}}[i,j] = \tilde{\boldsymbol{A}}_{\boldsymbol{c\to p}}[i, \boldsymbol{\delta}[i,j]]$
    end for
end for
for $j = 0, ..., N-1$ **do**
    $\tilde{\boldsymbol{A}}_{\boldsymbol{p\to c}}[:,j] = \boldsymbol{K_c}[j,:]\boldsymbol{Q_r}^\intercal$
end for
for $j = 0, ..., N-1$ **do**
    for $i = 0, ..., N-1$ **do**
        $\boldsymbol{A_{p\to c}}[i,j] = \tilde{\boldsymbol{A}}_{\boldsymbol{p\to c}}[\boldsymbol{\delta}[j,i], j]$
    end for
end for
$\tilde{\boldsymbol{A}} = \boldsymbol{A_{c\to c}} + \boldsymbol{A_{c\to p}} + \boldsymbol{A_{p\to c}}$
$\boldsymbol{H_o} = \text{softmax}(\frac{\tilde{\boldsymbol{A}}}{\sqrt{3d}})\boldsymbol{V_c}$

Output: $\boldsymbol{H_o}$

Enhanced Mask Decoder (EMD)

In the MLM task, it is useful to have some information on the absolute positions of tokens in the sequence to distinguish ambiguity found in relative position comparisons. The author's provide the example "a new **store** opened beside the new **mall**." In this sentence, masking *store* and *mall* does not provide any bidirectional context from relative position (as "new" is present around both masks). So, the BERT authors introduced absolute position in the input layer [1]. However, the DeBERTa authors have found delaying this inclusion until just before the softmax used to determine masked predictions allows for overall better performance.

Performance

DeBERTa's novel approach to attention augmentation allows for better performance on GLUE tasks [6]:

Model	CoLA Mcc	QQP Acc	MNLI-m/mm Acc	SST-2 Acc	STS-B Corr	QNLI Acc	RTE Acc	MRPC Acc	Avg.
BERT$_{large}$	60.6	91.3	86.6/-	93.2	90.0	92.3	70.4	88.0	84.05
RoBERTa$_{large}$	68.0	92.2	90.2/90.2	96.4	92.4	93.9	86.6	90.9	88.82
XLNet$_{large}$	69.0	92.3	90.8/90.8	**97.0**	92.5	94.9	85.9	90.8	89.15
ELECTRA$_{large}$	69.1	**92.4**	90.9/-	96.9	92.6	95.0	88.0	90.8	89.46
DeBERTa$_{large}$	**70.5**	92.3	**91.1/91.1**	96.8	**92.8**	**95.3**	**88.3**	**91.9**	**90.00**

Table 1: Comparison results on the GLUE development set.

And outperforms many other competitive models on natural language inference (NLI) tasks:

Model	MNLI-m/mm Acc	SQuAD v1.1 F1/EM	SQuAD v2.0 F1/EM	RACE Acc	ReCoRD F1/EM	SWAG Acc	NER F1
BERT$_{large}$	86.6/-	90.9/84.1	81.8/79.0	72.0	-	86.6	92.8
ALBERT$_{large}$	86.5/-	91.8/85.2	84.9/81.8	75.2	-	-	-
RoBERTa$_{large}$	90.2/90.2	94.6/88.9	89.4/86.5	83.2	90.6/90.0	89.9	93.4
XLNet$_{large}$	90.8/90.8	95.1/89.7	90.6/87.9	85.4	-	-	-
Megatron$_{336M}$	89.7/90.0	94.2/88.0	88.1/84.8	83.0	-	-	-
DeBERTa$_{large}$	**91.1/91.1**	**95.5/90.1**	**90.7/88.0**	**86.8**	**91.4/91.0**	**90.8**	**93.8**
ALBERT$_{xxlarge}$	90.8/-	94.8/89.3	90.2/87.4	86.5	-	-	-
Megatron$_{1.3B}$	90.9/91.0	94.9/89.1	90.2/87.1	87.3	-	-	-
Megatron$_{3.9B}$	91.4/91.4	95.5/90.0	91.2/88.5	89.5	-	-	-

Table 2: Results on MNLI in/out-domain, SQuAD v1.1, SQuAD v2.0, RACE, ReCoRD, SWAG, CoNLL 2003 NER development set. Note that missing results in literature are signified by "-".

Even base model performance is very competitive:

Model	MNLI-m/mm (Acc)	SQuAD v1.1 (F1/EM)	SQuAD v2.0 (F1/EM)
RoBERTa$_{base}$	87.6/-	91.5/84.6	83.7/80.5
XLNet$_{base}$	86.8/-	-/-	-/80.2
DeBERTa$_{base}$	**88.8/88.5**	**93.1/87.2**	**86.2/83.1**

Table 3: Results on MNLI in/out-domain (m/mm), SQuAD v1.1 and v2.0 development set.

Finally, performance on SuperGLUE [4] reveals DeBERTa$_{large}$ is the first model to outperform human baselines. This

was done by implementing a virtual adversarial training regimen called Scale-invariant Fine Tuning (SiFT) by perturbing input embeddings (disentangled context vectors) then *normalizing* into stochastic vectors. Normalizing using SiFT has proven to outperform standard aadversarial training methods as this process, only applied to fine-tune on SuperGLUE tasks, achieved impressive results:

Model	BoolQ Acc	CB F1/Acc	COPA Acc	MultiRC F1a/EM	ReCoRD F1/EM	RTE Acc	WiC Acc	WSC Acc	Average Score
RoBERTa$_{large}$	87.1	90.5/95.2	90.6	84.4/52.5	90.6/90.0	88.2	69.9	89.0	84.6
NEXHA-Plus	87.8	94.4/96.0	93.6	84.6/55.1	90.1/89.6	89.1	74.6	93.2	86.7
T5$_{11B}$	91.2	93.9/96.8	94.8	88.1/63.3	94.1/93.4	92.5	76.9	93.8	89.3
T5$_{11B}$+Meena	**91.3**	**95.8/97.6**	97.4	88.3/63.0	94.2/93.5	92.7	**77.9**	95.9	90.2
Human	89.0	95.8/98.9	100.0	81.8/51.9	91.7/91.3	93.6	80.0	100.0	89.8
DeBERTa$_{1.5B}$+SiFT	90.4	94.9/97.2	96.8	**88.2/63.7**	**94.5/94.1**	**93.2**	76.4	**95.9**	89.9
DeBERTa$_{Ensemble}$	90.4	95.7/97.6	**98.4**	88.2/63.7	94.5/94.1	93.2	77.5	95.9	**90.3**

Table 4: SuperGLUE test set results scored using the SuperGLUE evaluation server. All the results are obtained from https://super.gluebenchmark.com on January 6, 2021.

Notes

As of April 22, 2024, the Microsoft team has released 3 versions of DeBERTa, each better and larger than the previous. DeBERTa-v3 could still be considered a state-of-the-art LLM, while v1 seems to perform very poorly without fine-tuning, relative to similarly sized models currently.

In practical usage, DeBERTa uses significantly more memory than BERT and RoBERTa due to the disentangled positional encoding and word embeddings and, consequently, disentangled attention, despite optimization efforts. This may prove to be a considerable problem for scaling up model parameters or re-implementation of this feature in larger, SOTA models like GPT-4, Claude Opus, etc. (especially as the memory requirement scales faster than computational speed with parameter scaling).

Finally, although DeBERTa's attention formulation produces the 4 component disentangled attention matrices, the position-position matrix is not summed for usage in generating contextualized vectors. This seems like an oversight as although the authors state they are not concerned with absolute position during model training, it may help to include it regardless - at least on some attention heads. This component's inclusion could help the model to learn to solve ambiguous problems that require some sense of absolute position (or, equivalently, comparison of 2 or more relative position windows). Take the example of the sentence "a new store opened beside the new mall" as evidence.

Contribution

The authors of the DeBERTa paper present a novel idea but one that seems almost implicit in nature. As students, learning about the transformer architecture, we are taught the positional encoding is necessary to understand some lexical, positional information about the inputs and thus must be summed to the word embeddings. However, the question that has been asked in at least every class that has covered this topic is "why sum?" A simple yet interesting question that is purported by this paper by, not students, but professionals experimenting on a simple idea that has vast implications in the field.

Additionally, not only did the authors provide comparative metrics on GLUE and other NLI tasks, the author's provided stats on SuperGLUE - a relatively newer benchmark whih many models fail to meet and remain competitive on. Furthermore, creating the first model that outperforms human baselines on a respected benchmark is a testament to the potential of this novel idea and is conveyed in a respectively high manner. Finally, the description of disentangled attention and the EMD was clear and concise and addressed the primary concerns including complexity and efficiency.

Clarifications

The implementation of the disentangled attention is incredibly important to understand. The authors suggest to keep positional encoding and word embeddings separate and taking their dot product to produce 4 distinct attention components. However, the author's specifically omit the inclusion of the position-position component on the basis of only requiring relative positional context which is captured in the other 3 components. It is important to understand this difference as summing the position-position component could cause the model to over-attend to position-position relationships in the data with equal proportion to context-context relationships. This is crucial to allow the model's training regimen to put more emphasis on contextual features of the inputs than syntactical information.

Another possible area of misunderstanding is the Enhanced Mask Decoder (EMD). DeBERTa and other BERT-based models, including RoBERTa, are trained on scraped information from the internet (usually Wikipedia) specifically on masked language modeling. The specific structure of this task, "given a sequence $\boldsymbol{X} = \{x_i\}$, is to corrupt it into $\tilde{\boldsymbol{X}}$ by masking 15% of its tokens at random and then train the model parameterized by θ to reconstruct $\boldsymbol{X}$ by predicting the masked tokens $\tilde{x}$ conditioned on $\tilde{\boldsymbol{X}}$:"

$$\max_{\theta} \log p_\theta(\boldsymbol{X}|\tilde{\boldsymbol{X}}) = \max_{\theta} \sum_{i \in \mathcal{C}} \log p_\theta(\tilde{x}_i = x_i|\tilde{\boldsymbol{X}})$$

"where $\mathcal{C}$ is the index set of the masked tokens in the sequence. Traditionally, "the authors of BERT propose to keep 10% of the masked tokens unchanged, another 10% replaced with randomly picked tokens and the rest replaced with the `[MASK]` token."
Consequently, EMD proposes including absolute positional data as a vector sum just before softmax to predict the classification of the mask instead of at the input time (as BERT and Roberta do). This simple change, which can be easily overlooked, has a significant impact on the performance of DeBERTa, which is reflected in the SuperGLUE benchmark (it is crucial to point out that EMD is only applied for fine-tuning DeBERTa on SuperGLUE and no other task - all other benchmarks are using the standard DeBERTa models).

Misgivings

Although the authors present quite compelling metrics and results in the paper, they fail to mention the lacking performance of the base DeBERTa model. Experimenting with DeBERTa$_base$ on GLUE, SQuAD, and a fine-tuned test set, the standard model heavily underperforms relative to BERT and RoBERTa on similar aspects. In fact, experimenting with the Inference API for DeBERTa-v1 hosted on HuggingFace (), we find the performance of MLM on unseen data and examples shows incredibly poor performance, even considering the top-10 results. However, after fine-tuning the model on the specific task, we find much better performance across the board, though it is important to say that fine-tuning DeBERTa over the same training set takes much longer and requires more memory (for the reasons mentioned previously) as compared to the competing BERT-based models. In the presentation, I show the results of these models on standard benchmarks alongside this paper.

Future Works

Iterative versions of DeBERTa mitigate most of the misgivings mentioned earlier. As mentioned earlier, DeBERTa-v3 performs very well and is on par with similarly sized encoders. That said, the architecture of the model remains nearly the same. So, it would be nice to see a model that incorporates *all* 4 components of the derived disentangled attention to evaluate performance relative to the current DeBERTa implementation. Similarly, it would be interesting to see how SiFT performs on benchmarks and tasks besides SuperGLUE to determine whether the adversarial training regimen generalizes to natural language tasks beyond NLI.

DeBERTa's disentangled attention also allows for an easier time picking apart the model parameters towards a more explainable description of the model. It would be beneficial for the reader to take a look at Chain-of-Thought examples on DeBERTa and other mechanistic interpretability work on DeBERTa as compared to other BERT-based models. Furthermore, the inclusion of SiFT in this paper opens the doors for comparison of adversarial perturbation techniques and defenses against such attacks. Specifically, the exploration of language-based GANs and the performance of DeBERTa+GAN would be very interesting to see as compared with simply DeBERTa+SiFT/DeBERTa+EMD.

Industrial Applications

DeBERTa's disentangled attention mechanism offers practical solutions for various industries. In finance, it can enhance algorithmic trading strategies by analyzing market news and social media sentiment with a deeper understanding of context and positional awareness of time series data, leading to more informed investment decisions (currently, Morgan Stanley has been known to use sentiment analysis models; however, it is unknown if this model incorporates disentangled attention -). In healthcare, DeBERTa's capabilities can aid in medical coding and billing, ensuring accuracy and efficiency in healthcare administration. Customer service platforms can deploy DeBERTa to improve chat-bot interactions, providing personalized responses and resolving customer inquiries more effectively. Legal firms can leverage DeBERTa for

contract review and due diligence, automating tedious tasks and reducing human error. Additionally, DeBERTa can optimize inventory management in manufacturing and logistics by analyzing supply chain documents and predicting demand fluctuations. These practical applications, although shared by other encoder models and most LLMs, require more contextual awareness, for which DeBERTa is unparalleled in overall comprehension for its size.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. 2019.

[2] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. 2021.

[3] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. 2019.

[4] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. 2020.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. 2023.

[6] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. 2019.