Poly-encoders: architectures and pre-training strategies for fast and accurate multi-sentence scoring

Link: https://arxiv.org/pdf/1905.01969.pdf

Biencoders cached the encoded inputs, Cross Encoder must recompute encoding for each input.

Poly-encoder has an additional learnt attention mechanism that represents **more global features from which to perform self-attention**.

Related Work

- In Cross Encoder, there is concatenation of the input and a candidate serve as a new input to a nonlinear function that scores their match based on any dependencies it wants.
    - applying self-attention at every layer.
    - every word at candidate can interact with every word at input

Task

- ConvAI2 and Ubuntu V2 task is about conversations between two personas, model will have to give a response to a question out of twenty choices, the best model has given an accuracy of about 80
- Wiki Article Search, given a sentence, find the article. Almost like a web search.

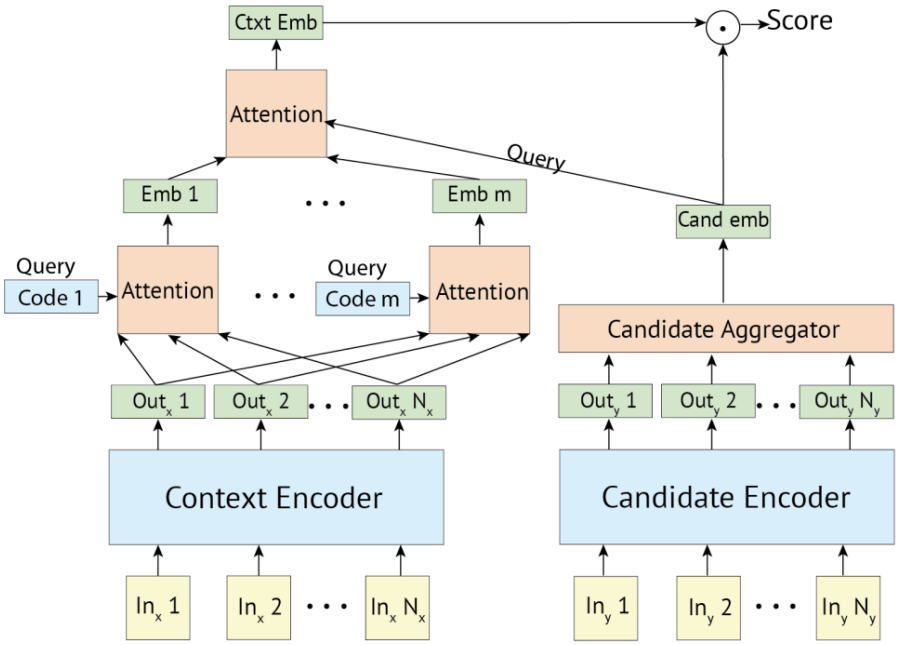
Methods

- Transformers, 2 bert-base like models were trained, one was to check if **reproducing training like BERT gives same performance (case 1)**, and another way was to check **if pretraining on data that will be used in downstream is helpful or not (case 2)**.
- Input Represented by [Input,Label], The input was generally followed by a next sentence, or next utterances in a dialogue.

- Pretraining had MLM, NSP and NUP (next utterance prediction, can be many sentences), it had 50% positive and 50% negative next sentences/utterance.
- In BiEncoder, both transformers gives a sequences of vectors, which can be combined in a single representation of input and candidate,
    - average of all vectors, or average of all vectors upto m, or first output of the transformer (inputs are surrounded by [S] ⇒ first output)
    - scores are dot product of context and candidate,
- Cross Encoder, input jointly encoded for better representation, the input is surrounded by [S], cross attention over this input helps in rich extraction mechanism,
    - linear layer applied over embedding,
        - linear layer performs a weighted sum of the input features followed by an activation function (such as ReLU or Tanh) to introduce non-linearity
        - **score calculation would be like projection of hidden states of last layer corresponding to CLS token into a single value, then a sigmoid is applied to reduce it to 0-1.**

$$Rel(user, ads) = \sigma(H_{[CLS]} W_A),$$

    - no precomputed embeddings, and hence it takes longer time, because of one transformer and input and candidate combined.
- In Poly Encoder, candidate is encoded as one vector (BE),
    - it will have a precomputed cache of input responses, to understand longer contexts, it is split into m sequences and mapped using context codes, using context codes and intermediate representation of previous sequences, we can understand the stories better.
    - y1 will focus on important contexts of the first part, y2 will focus on important context of second part of context, so on.
    - **generate attention over M embeddings and candidate embeddings to get scores.**
    - In training the larger the number of context codes, the better.



Domain Specific PT

1. The study fine-tunes two different transformer models, one pre-trained on Reddit data and the other on Toronto Books + Wikipedia data, for four specific tasks.
2. When using the pre-trained weights from Reddit data, they optimize all layers of the transformer with the Adamax optimizer, and they re-scaled the last linear layer to match the standard deviation of BERT's output for better results.
3. Fine-tuning with the Reddit-pre-trained transformer leads to state-of-the-art performance in all three dialogue tasks, indicating that pre-training with a similar task to the downstream tasks (dialogue data for dialogue tasks) is beneficial **(case 2)**, as seen in previous research. Fine-tuning with Toronto Books + Wikipedia data also yields similar results to the original BERT weights **(case 1)**.