Constrained Beam Search

1 Idea

We first obtain the list of all english tokens by decoding the tokens into string and checking the ASCII values of the characters. We maintain two separate beams of b length. The first beam is like the normal beam in vanilla Beam search Algorithm call this beam "**vanilla_beam**" while in the other beam all the sequence have the invariant that they contain atleast one english token we call this beam "**english_beam**". In every update step, we compute 3 list of sequences, L1, L2, L3.

- **L1**: The sequences obtained by adding the top (b+1) tokens into each vanilla sequence.
- **L2**: The sequences obtained by adding the top (b+1) **english** tokens into each vanilla sequence.
- **L3**: The sequences obtained by adding the top (b+1) tokens into each english sequence.

We update the beams as:

- **english_beam**: Best b sequences in $(L2 \cup L3)$.
- **vanilla_beam**: Best b sequences in $L1$.

Whenever a $< eot >$ gets appended into a sequence of the **english_beam** we add it to the finished list. Also in the finalize function we only consider the english beams.

2 Algorithm

2.1 Update

Algorithm 1 Update Function for Constrained Beam Search

Input: *tokens*, *logits*, *sum_logprobs*
// Initialization
Initialize lists $L1$, $L2$, $L3$ as empty
for ray in vanilla_beam **do**
    $l1 \leftarrow$ best $k+1$ tokens for the ray
    for each token in $l1$ add ray.concat(token) to $L1$
    $l2 \leftarrow$ best $k+1$ English tokens for the ray
    for each token in $l2$ add ray.concat(token) to $L2$
end for
for ray in english_beam **do**
    $l3 \leftarrow$ best $k+1$ tokens for the ray
    for each token in $l3$ add ray.concat(token) to $L3$
end for
$L_{\text{eng}} \leftarrow$ top $k+1$ sequences of $(L2 \cup L3)$
$L_{\text{all}} \leftarrow$ top $k+1$ sequences of $L1$
if any ray in L_{all} is completed (token is *eot*) **then**
    pop the ray from L_{all}
end if
if any ray in L_{eng} is completed (token is *eot*) **then**
    pop the ray from L_{eng} and add to $finished\_sequences$
end if
Set vanilla_beam to top k elements from L_{all}
Set english_beam to top k elements from L_{eng}
if size of finished is *max_candidates* **then**
    completed $\leftarrow$ true
else
    completed $\leftarrow$ false
end if
return vanilla_beam, english_beam, completed