NPFL116 Compendium of Neural Machine Translation

Model Ensembling and Beam Search

March 22, 2017

Jindřich Helcl, Jindřich Libovický

Charles Univeristy in Prague
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics

Greedy Decoding

- In each step, the model computes a distribution over the vocabulary V (given source $\mathbf{x}$, the previous outputs h, and the model parameters θ).

$$\mathrm{p}(y|h) = g(\mathbf{x}, h, \theta)$$

- In greedy decoding:

$$y^* = \operatorname*{argmax}_{y \in V} \mathrm{p}(y|h)$$

- Repeat, until an end-of-sentence symbol (</s>) is decoded.

Greedy Decoding — cont.

- Pros:
  - Fast and memory-efficient
  - Gives reasonable results
- Cons:
  - We are interested in the most probable *sentence*:

$$(y^*)_{i=0}^{N} = \operatorname*{argmax}_{(y)_{i=0}^{N}} \mathrm{p}(y_0, \ldots, y_N | h)$$

  - Other methods: *much better* results for the cost of a slow-down.

Model Ensembling

- Combine word probabilities from M models:

$$\mathrm{p}(y|h) = \bigoplus_{m=0}^{M} p(y|h, \theta_m)$$

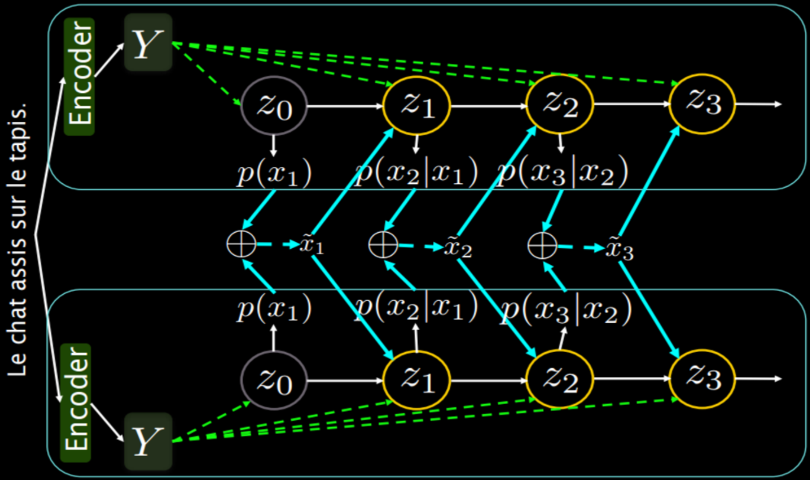
- The additive function $\bigoplus$:
  - Majority voting scheme (arithmetic mean):

$$\bigoplus_{m=0}^{M} \mathbf{f}(x) = \frac{1}{M} \sum_{m=0}^{M} f_m(x)$$

  - Consensus building scheme (geometric mean):

$$\bigoplus_{m=0}^{M} \mathbf{f}(x) = \sqrt[M]{\prod_{m=0}^{M} f_m(x)}$$

Model Ensembling — picture



Source: `https://nlp.stanford.edu/projects/nmt/Luong-Cho-Manning-NMT-ACL2016-v4.pdf`

Beam Search

- Instead of taking the $\arg\max$ in every, step, keep a list (or *beam*) of k-best scoring *hypotheses*.
- Hypothesis = partially decoded sentence $\rightarrow$ score
- Hypothesis score $\psi_t = (y_1, y_2 \ldots, y_t)$ is the probability of the decoded sentence prefix up to t-th word.

$$\mathrm{p}(y_1, \ldots, y_t | h) = \mathrm{p}(y_1 | h) \cdot \cdots \cdot \mathrm{p}(y_t | y_1, \ldots, y_{t-1} | h)$$

- Rule to compute the score of an *extended* hypothesis ψ_t:

$$\mathrm{p}(\psi_t, y_{t+1} | h) = \mathrm{p}(\psi_t | h) \cdot \mathrm{p}(y_{t+1} | h)$$

- Prefers shorter hypotheses $\rightarrow$ normalization necessary.

Beam Search — algorithm

1. Begin with a single empty hypothesis in the beam.
2. In each time step:
   2.1 Extend all hypotheses in the beam by k most probable words (we call these *candidate hypotheses*)
   2.2 Sort the candidate hypotheses by their score.
   2.3 Put the best k hypotheses in the new beam.
   2.4 If a hypothesis from the beam reaches the end-of-sentence symbol, we move it to the list of finished hypotheses.
3. Finish (1) at the final time step or (2) all k-best hypotheses end with `</s>`
4. Sort the hypotheses by their score and output the best one.

Beam Search — picture

Closing Remarks

- The probabilities are never actually computed — everything is done in the logarithmic space.
- For probabilities p_1, p_2 and their logarithms e_1 and e_2, we use these rules:
  - Product of the probabilities becomes their sum in logarithmic space:

$$\log(p_1 \cdot p_2) = \log p_1 + \log p_2 = e_1 + e_2$$

  - Sum of the probabilities becomes the *log-sum-exp* in log-space:

$$\log(p_1 + p_2) = \log(\exp(e_1) + \exp(e_2)) = LSE(e_1, e_2)$$

  - Log-sum-exp function can be computed without any arithmetic operation in the linear space.
    For sorted sequence $e_1 > e_2 > \ldots > e_n$:

$$LSE(e_1, e_2, \ldots, e_n) = e_1 + LSE(e_2 - e_1, \ldots, e_n - e_1)$$