Cache-based Recurrent Transformer Network

概览

对于时间步 t，输入片段（向量）为 $X_t \in R^{L\times d}$，Memory 为 $\mathcal{M} = \{K_i, V_i\}_{i=1}^{N}$, 网络的更新步骤大致分为

查询

查询应该符合如下原则：

- 对于片段中的每个词，应有一个查询结果。
- 对于每个词的查询，这个词之后的词应该每 mask 住。
- 对于片段的第一个词，后面的词都是看不见的，查询时信息太少，应该拼接上一个片段 X_{t-1} 参与查询
- 从大小为 N 的缓存中查询中 k 个片段

$$\{\alpha_i, Z_i\}_k^{(j)} = Query(X_{t-1}, Masked_j(X_t), \mathcal{M}) \tag{1}$$

其中 Z_i 为需要进行回忆（以 Transformer Memory 的方式拼接进当前片段）的 k 个片段，α_i 为其对应的权重，j 表示计算第 j 个词对应的查询结果，$j = 1, 2, \cdots, L$
$\mathcal{M}$ 是 Memory，存储 N 个 Key-Value 对，Value 即为片段，对应的 Key 为该片段的一个意义向量。

更新 hidden state

使用 Transformer 网络更新 m 层隐状态

$$h_{t,j}^{1:m} = Transformer(\alpha_{1:k}^{(j)}, Z_{1:k}^{(j)}, X_t) \tag{2}$$

更新 Memory

$$\mathcal{M} = renew(h_t^{1:m}, \mathcal{M}) \tag{3}$$

1 查询

采取键值对匹配的方法，$Q_t^{(j)}$ 是对应的查询向量，计算方法列下，*Keys* 是缓存中的键向量。

$$\{\alpha_i, Z_i\}_k^{(j)} = topk(softmax(Q_t^{(j)} \cdot Keys^T))$$

以下列出查询向量 $Q_t^{(j)}$ 的计算方法：

- **1.1** fixed_length 定长方法: $Q_t^{(j)}$ 包含包括第 j 个词在内的第 j 个词之前的 L 个词（事实上已经取到 X_{t-1} 中，此处仍用 X_t 表示）
  - **1.1.1** fixed_length_1 直接使用底层的连续 L 个词，使用 encoder 进行计算

$$Q_t^{(j)} = (X_{t,j-L+1}, \cdots, X_{t,j-1}, X_{t,j})$$

$$Q_t^{(j)} = summary(Transformer(Q_t^{(j)}))$$

  - **1.1.2** fixed_length_2 将 j 个词作为当前片段，上一片段中的后 L-j 个词的表示作为 memory

$$Q_t^{(j)} = Transformer(X_{t-1}[j+1:L], X_t[1:j])$$

$$Q_t^{(j)} = summary(Q_t^{(j)})$$

  其中 *summary* 函数与之后更新 Memory 时使用的相同，*Transformer* 共享模型参数
- **1.2** 倍长方法：将 X_{t-1} 的 L 个词拼接入当前输入，使用 2L 个词进行计算

$$Q_t^{(j)} = concat(X_{t-1}, Masked_j(X_t))$$

$$Q_t^{(j)} = Transformer(Q_t^{(j)})$$

  注意到此时 $Q_t^{(j)}$ 为 2L 个词的表示，需要将其变成 L 个词的表示，才能使用 *summary* 函数
  - **1.2.1** last_l 截取后 L 个

$$Q_t^{(j)} = summary(Q_t^{(j)}[L+1:2L])$$

  - **1.2.2** middel_l 截取中间 L 个

$$Q_t^{(j)} = summary(Q_t^{(j)}[j+1:j+L])$$

  - **1.2.3** linear 线性变换法

$$Q_t^{(j)} = summary(Linear(Q_t^{(j)}))$$

  其中 *summary* 函数与之后更新 Memory 时使用的相同，*Transformer* 共享模型参数

2 更新 hidden state

- **2.1** standard：采用 Transformer-XL 的方法

$$h_{t,j}^{1:m} = Transformer(\alpha_{1:k}^{(j)}, Z_{1:k}^{(j)}, X_t)$$

对于 n=1，...，m

$$\mathbf{m}_{t,j}^{n-1} = concat(\{{Z^{(j)}}_i^{n-1}\}_{i=1}^{k})$$

$$\tilde{\mathbf{m}}_{t,j}^{n-1} = concat(\{\alpha_i^{(j)}{Z^{(j)}}_i^{n-1}\}_{i=1}^{k})$$

$$\tilde{\mathbf{h}}_{t,j}^{n-1} = \left[\mathrm{SG}\left(\mathbf{m}_{t,j}^{n-1}\right) \circ \mathbf{h}_t^{n-1}\right]$$

$$\hat{\mathbf{h}}_{t,j}^{n-1} = \left[\mathrm{SG}\left(\tilde{\mathbf{m}}_{t,j}^{n-1}\right) \circ \mathbf{h}_t^{n-1}\right]$$

$$\mathbf{q}_{t,j}^n, \mathbf{k}_{t,j}^n, \mathbf{v}_{t,j}^n = \mathbf{h}_{t,j}^{n-1}\mathbf{W}_q^{n\top}, \tilde{\mathbf{h}}_{t,j}^{n-1}{\mathbf{W}_{k,E}^n}^{\top}, \hat{\mathbf{h}}_{t,j}^{n-1}\mathbf{W}_v^{n\top}$$

$$\mathbf{A}_{t,j,i}^n = {\mathbf{q}_{t,j}^n}^{\top}\mathbf{k}_{t,i,j}^n + {\mathbf{q}_{t,j}^n}^{\top}\mathbf{W}_{k,R}^n\mathbf{R}_{i-j} + u^{\top}\mathbf{k}_{t,i,j} + v^{\top}\mathbf{W}_{k,R}^n\mathbf{R}_{i-j}$$

$$\mathbf{a}_{t,j}^n = \text{Masked-Softmax}\left(\mathbf{A}_{t,j}^n\right)\mathbf{v}_{t,j}^n$$

$$\mathbf{o}_t^n = \text{LayerNorm (Linear }(\mathbf{a}_t^n) + \mathbf{h}_t^{n-1})$$

$$\mathbf{h}_t^n = \text{Positionwise-Feed-Forward }(\mathbf{o}_t^n)$$

3 更新 Memory

Memory 的大小为 N 个 Key-Value 对，每个 Value 为一个片段 X_i 的所有层的表示。将上一步生成的 $h_t^{1:m}$ 存入 Memory:

- 将第一个 Key-Value 对删除，其余的向前递补，最后一个 Key-Value 对为空，将 $h_t^{1:m}$ 填入 Memory 中的最后一个空位

对于填入的 Key-Value 对 h_t，需要更新其 Key，用最顶层表示更新 Key：

$$Key = summary(h_t^n)$$

- **3.1 直接使用原向量** $summary(h_t^n) = Identical(flat(h_t^n))$
- **3.2** 线性变换 $summary(h_t^n) = Linear(flat(h_t^n))$