1 Project Specification

This project is intended to explore the many variety of Seq2Seq architecture. All the projects are build using AllenNLP framework.

2 Data

BBC Dataset

3 Pointer Generator Salience

This is our implementation of pointer generator that is repurposed for salience prediction.

4 Pointer Generator

This sub-project re-implement the paper of See, 2017.

4.1 Notes

1. All index start from 1, the index of 0 is reserved for initialization.

4.2 Input

4.2.1 Spec

Source sentence, $x_i \in \mathbb{R}^v$,

$$X = [x_1, \cdots, x_{T_s}]$$

, source salience, $z_i \in \{1, 0\}^f$

$$Z = [z_1, \cdots, z_{T_s}]$$

and target sentence, $y_j \in \mathbb{R}^v$,

$$Y = [y_1, \cdots, y_{T_t}]$$

where:

1. T_s and T_t are 400 and 100.
2. y_{T_t}, are BOS and EOS tokens.
3. v is the number of vocabulary.
4. f is the number of feature

4.2.2 Review

1. `data.py`
   (a) `article2ids`: Convert words in article into indexes, in addition extend the vocabulary by the article' tokens that are not in vocabulary. Return ids (the article indexes) and oovs (extra word list that is not in vocabulary). Note oovs can be used to retrieve the token from the extended vocab index.
   (b) `abstract2ids`: Convert words in abstract into indexes with little caveat: the oov that exist in the article extended vocab is used instead of the unknown token.
   (c) `outputids2words`: Retrieving token based on index with mapping support for the extended vocabulary

4.3 Encoder

4.3.1 Spec

The inputs are passed to LSTM to produce $\mathbf{h}_i$ and $\mathbf{c}_{i-1}$ which is an encoder hidden and context state at time-step i. There are three variants for the input encoder:

1. The standard Encoder

$$\mathbf{h}_i \in \mathbb{R}^{2h}, \mathbf{c}_i \in \mathbb{R}^{2h} = \text{LSTM}(e_s(x_i) \in \mathbb{R}^E, (\mathbf{h}_{i-1}, \mathbf{c}_{i-1}))$$

2. Salience as Raw

$$\mathbf{h}_i \in \mathbb{R}^{2h}, \mathbf{c}_i \in \mathbb{R}^{2h} = \text{LSTM}([e_s(x_i), z_i] \in \mathbb{R}^{(E+f)}, (\mathbf{h}_{i-1}, \mathbf{c}_{i-1}))$$

3. Salience as Embedding

$$\mathbf{h}_i \in \mathbb{R}^{2h}, \mathbf{c}_i \in \mathbb{R}^{2h} = \text{LSTM}([e_s(x_i), e_z(z_i)] \in \mathbb{R}^{(E+f)}, (\mathbf{h}_{i-1}, \mathbf{c}_{i-1}))$$

   where: $e_z(z_i) = \tanh(z_i(\mathbf{W}^{f\times E})^\intercal + \mathbf{b}^E)$

4. Salience as Attended Embedding using Bilinear Attention

$$\mathbf{h}_i \in \mathbb{R}^{2h}, \mathbf{c}_i \in \mathbb{R}^{2h} = \text{LSTM}(\mathbf{f}_i \in \mathbb{R}^E, (\mathbf{h}_{i-1}, \mathbf{c}_{i-1}))$$

   where:

   (a) $\mathbf{f} = \mathbf{P}^\intercal \mathbf{f}'$
   where $\mathbf{P} \in \mathbb{R}^{(K\times E)}$

   (b) $\mathbf{f}'_i = \text{BAN}(e_s(x_i), e_z(z_i))$

   (c) $\mathbf{f}'^k_i = (e_s(x_i)^\intercal \mathbf{U}')^\intercal_k \mathcal{A}(e_z(z_i)^\intercal \mathbf{V}')_k$
   where $\mathbf{U}' \in \mathbb{R}^{E\times K}$, $\mathbf{V}' \in \mathbb{R}^{f\times K}$,$(e_s(x_i)^\intercal \mathbf{U}') \in \mathbb{R}$ and $(e_z(z_i)^\intercal \mathbf{V}') \in \mathbb{R}^f$

   (d) $\mathcal{A} = \text{softmax}((((\mathbb{1} \cdot \mathbf{p}^\intercal) \circ e_s(x_i)^\intercal \mathbf{U})(\mathbf{V}^\intercal e_z(z_i)))$
   where $\mathbf{p} \in \mathbb{R}^K$

Since it is a bidirectional LSTM, the hidden is as such.

$$\mathbf{h}_i = [\overleftarrow{\mathbf{h}_i}, \overrightarrow{\mathbf{h}_i}]$$
$$\overleftarrow{\mathbf{h}_i}, \overrightarrow{\mathbf{h}_i} \in \mathbb{R}^h,$$

The final state and context are reduced to half their dimension.

$$\mathbf{h}'_{T_s} = \text{ReLU}(\mathbf{h}_{T_s}(\mathbf{W}^{H\times 2H})^\intercal)$$
$$\mathbf{c}'_{T_s} = \text{ReLU}(\mathbf{c}_{T_s}(\mathbf{W}^{H\times 2H})^\intercal)$$

4.3.2 Review

1. `model.py`: `Encoder` and `ReduceState`

4.4 Decoder

4.4.1 Spec

The final encoder hidden is passed to decoder where $\mathbf{s}_j$ is a decoder state at time-step j where during generation:

$$\mathbf{s}_j \in \mathbb{R}^H = \text{LSTM}(\mathbf{s}_{j-1}, \mathbf{e}^*_j)$$

where:

1. e_s and e_t are sharing weight.
2. $\mathbf{s}_0 = (\mathbf{h}'_{T_s}, \mathbf{c}'_{T_s})$.
3. $\mathbf{h}^*_0 = 0$
4. $\hat{y}_j = y_{j-1}$, teacher forcing.
5. $\mathbf{e}^*_\mathbf{j} = [e_t(\hat{y}_j); \mathbf{h}^*_j](\mathbf{W}^{(2H+i)\times i})^\intercal$

The output is as follows.

$$\mathbf{o}_j = ([\mathbf{s}_j; \mathbf{h}^*_j](\mathbf{W}^{H\times 3H})^\intercal + b)(\mathbf{W}^{H\times V})^\intercal + b$$

Distribution is:

$$p(y_j|y_{<j}, x_{1:T_s}) = \text{Softmax}(\mathbf{o}_j)$$

4.4.2 Review

`model.py`: `Decoder` class

4.5 Attention

4.5.1 Spec

A dynamic fixed length vector (context vector with attention),

$$\mathbf{h}_j^* \in \mathbb{R}^{2H} = \sum_{i=1}^{T_s} \alpha_{ij} \mathbf{h}_i$$

The α_{ij} is the attention probability that is determined by:

$$\alpha_{ij} = \frac{e^{\eta(\mathbf{s}'_j, \mathbf{cov}_i^j, \mathbf{h}_i)}}{\sum_{i'} e^{\eta(\mathbf{s}'_j, \mathbf{cov}_{i'}^j, \mathbf{h}_{i'})}}$$

where:

1. The coverage is determined by:

$$\mathbf{cov}_i^j = \sum_{j'=0}^{j-1} a_i^{j'}$$

   and

$$\mathbf{cov}_i^0 = 0$$

2. $\mathbf{s}'_0 \in \mathbb{R}^{2h} = [\mathbf{h}'_{T_s}; \mathbf{c}'_{T_s}]$
3. The scoring function $\eta(\cdot)$ is as follows.

$$\eta(\mathbf{s}'_j, \mathbf{cov}_i^j, \mathbf{h}_i) = ([\mathbf{s}'_j; \mathbf{cov}_i^j; \mathbf{h}_i](\mathbf{W}^{2h \times 2h})^\intercal + \mathbf{b})(\mathbf{v}^{1 \times 2h})^\intercal$$

4.6 Loss Function

The class distribution is determined as follows.

$$P(w) = p_{\text{gen}} P_{\text{vocab}}(w) + (1 - p_{\text{gen}}) \sum_{i:w_i=w} a_i^t$$

where:

$$p_{\text{gen}} = \sigma([\mathbf{h}_j^*; \mathbf{s}_j; \mathbf{e}_j^*](\mathbf{W}^{1 \times (3h+i)})^\intercal + b)$$

4.6.1 Math

5 CopyNet

CopyNet from https://arxiv.org/pdf/1603.06393.

5.1 Model details

Source sentence,

$$X = [x_1, \cdots, x_{T_s}]$$

to a dynamic fixed length vector (context vector with attention),

$$\mathbf{c}_j = \sum_{i=1}^{T_s} \alpha_{ij} \mathbf{h}_i$$

$$\alpha_{ij} = \frac{e^{\eta(\mathbf{s}_{j-1}, \mathbf{h}_i)}}{\sum_{i'} e^{\eta(\mathbf{s}_{j-1}, \mathbf{h}_{i'})}}$$

where $\mathbf{h}_i$ is an encoder state at timestep i

$$\mathbf{h}_i = f_s(e_s(x), \mathbf{h}_{i-1})$$

and $\mathbf{s}_j$ is a decoder state at timestep j where during generation:

$$\mathbf{s}_j = f_t(e_t(y_{j-1}), \mathbf{s}_{j-1}, \mathbf{c}_j)$$

where

1. f_s is a bi-LSTM for source sequence.
2. f_t is an LSTM for target sequence.
3. e_s and e_t are embedding for source and target sequence.
4. η is a scoring function using MLP.

and during copying:

$$\mathbf{s}_j = f_t([e_t(y_{j-1}); \zeta(y_{j-1})], \mathbf{s}_{j-1}, \mathbf{c}_j)$$

where

$$\zeta y_{j-1} = \sum_{i=1}^{T_s} \rho_{ij} \mathbf{h}_i$$

and

$$\rho_{ij} = \begin{cases} \frac{1}{K} p_c(x_i|\mathbf{s}_j, \mathbf{M}), & x_i = y_{j-1} \\ 0, & \text{otherwise} \end{cases}$$

where K is a normalization for x_i. The need of another normalization is due to the possibility of having several duplicates with the same probability.

The initial encoder, and decoder state are

$$\mathbf{h}_0 = 0 \in \mathbb{R}^{D_h}$$
$$\mathbf{s}_0 = \mathbf{h}_{T_s} \in \mathbb{R}^{D_h}$$

where D_h is a hidden dimension.

The predicted target symbol y_j is sampled from a distribution or copied from source text

$$p(y_j|y_{\leq j}, X) = p_g(y_j|y_{\leq j}, X) + p_c(y_j|y_{\leq j}, X)$$

where

1. p_g and p_c are the distribution of generate and copy mode.
2. g is an output function (MLP and softmax).

The distribution of generate mode is

$$p_g(y_j|y_{\leq j}, X) = \begin{cases} \frac{1}{\mathbf{Z}} e^{\psi_g(y_j)}, & y_j \in \mathcal{V} \\ 0, & y_t \in \mathcal{X} \cap \bar{\mathcal{V}} \\ \frac{1}{\mathbf{Z}} e^{\psi_g(\text{UNK})}, & y_t \notin \mathcal{X} \cap \mathcal{V} \end{cases}$$

The distribution of copy mode is

$$p_c(y_j|y_{\leq j}, X) = \begin{cases} \frac{1}{\mathbf{Z}} \sum_{i:x_i=y_j} e^{\psi_c(x_i)}, & y_j \in \mathcal{X} \\ 0, & \text{otherwise} \end{cases}$$

where

1. ψ_g and ψ_c are scoring function for generate and copy mode.
2. Z is a normalization $Z = \sum_{v \in \mathcal{V} \cup \{UNK\}} e^{\psi_g(v)} + \sum_{x \in \mathcal{X}} e^{\psi_c(x)}$

The scoring function for generate and copy mode are:

$$\psi_g(y_j = v_k) = \mathbf{v_k}^\intercal \mathbf{W}_o \mathbf{s_j}, \ v_k \in \mathcal{V} \cup \text{UNK}$$

where

1. $\mathbf{W}_o \in \mathbb{R}^{(N+1)\times H_d}$
2. $\mathbf{v}_k$ is a one-hot encoding for v_k.

$$\psi_c(y_j = x_k) = \sigma(\mathbf{h}_k^\intercal \mathbf{W}_c)\mathbf{s_j},\ x_k \in \mathcal{X}$$

where

1. $\mathbf{W}_c \in \mathbb{R}^{H_d \times H_d}$
2. σ is a non-linear activation function (tanh)

5.2 Implementation

The following is the implementation guideline from AllenNLP copynet model.

5.2.1 CopyNetDatasetReader

The following are the outputs of the custom reader:

1. source tokens
2. source token ids which is a separate indexing scheme that use the document+target vocabulary (instance specific) in contrast with source tokens which use global vocabulary. Target token ids is the reverse of source token ids.
3. source to target is the mapping which use the vocabulary of target namespace to index the source (build using Namespace swapping field).
4. metadata is the text sequence from the source and target (build using Metadata field).

Technical noteworthy:

1. NamespaceSwappingField takes tokens and map it using another namespace.