Seq2seq model

Deep Learning Lecture 9

Samuel Cheng

School of ECE
University of Oklahoma

Spring, 2018
(Slides credit to Stanford CS20si)

Table of Contents

Overview

- We will look into the sequence-to-sequence model. We will look into two examples
  - Neural machine translation (NMT)
  - Chatbot

Sequence-to-sequence model
Choi et al. 2014

- The current model class of choice for most dialogue and machine translation systems
- Introduced by *Cho et al*. in 2014 (from Bengio's group) for Statistical Machine Translation, the predecessor of neural-machine translation (NMT)
  - Both Google and Microsoft have the translation service has switched to NMT-based system since November 2016
- The paper "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation" has been cited 2,200 times, over one time a day.
- Originally called "RNN Encoder-Decoder"

Sequence-to-sequence model
Choi et al. 2014

- The current model class of choice for most dialogue and machine translation systems
- Introduced by *Cho et al*. in 2014 (from Bengio's group) for Statistical Machine Translation, the predecessor of neural-machine translation (NMT)
  - Both Google and Microsoft have the translation service has switched to NMT-based system since November 2016
- The paper "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation" has been cited 2,200 times, over one time a day.
- Originally called "RNN Encoder-Decoder"

Sequence-to-sequence model
Choi et al. 2014

- The current model class of choice for most dialogue and machine translation systems
- Introduced by *Cho et al*. in 2014 (from Bengio's group) for Statistical Machine Translation, the predecessor of neural-machine translation (NMT)
  - Both Google and Microsoft have the translation service has switched to NMT-based system since November 2016
- The paper "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation" has been cited 2,200 times, over one time a day.
- Originally called "RNN Encoder-Decoder"

Sequence-to-sequence model
Choi et al. 2014

- The current model class of choice for most dialogue and machine translation systems
- Introduced by *Cho et al*. in 2014 (from Bengio's group) for Statistical Machine Translation, the predecessor of neural-machine translation (NMT)
  - Both Google and Microsoft have the translation service has switched to NMT-based system since November 2016
- The paper "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation" has been cited 2,200 times, over one time a day.
- Originally called "RNN Encoder-Decoder"

Sequence-to-sequence model
Choi et al. 2014

Consists of two RNNs:

- Encoder maps a variable-length source sequence (input) to a fixed-length vector
- Decoder maps the vector representation back to a variable-length target sequence (output)
- Two RNNs are trained jointly to maximize the conditional probability of the target sequence give a source sequence

Sequence-to-sequence model
Choi et al. 2014

Consists of two RNNs:

- Encoder maps a variable-length source sequence (input) to a fixed-length vector
- Decoder maps the vector representation back to a variable-length target sequence (output)
- Two RNNs are trained jointly to maximize the conditional probability of the target sequence give a source sequence

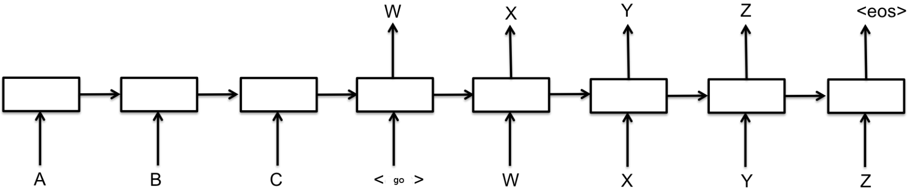
Sequence-to-sequence model
Choi et al. 2014

Consists of two RNNs:

- Encoder maps a variable-length source sequence (input) to a fixed-length vector
- Decoder maps the vector representation back to a variable-length target sequence (output)
- Two RNNs are trained jointly to maximize the conditional probability of the target sequence give a source sequence

Simplest architecture

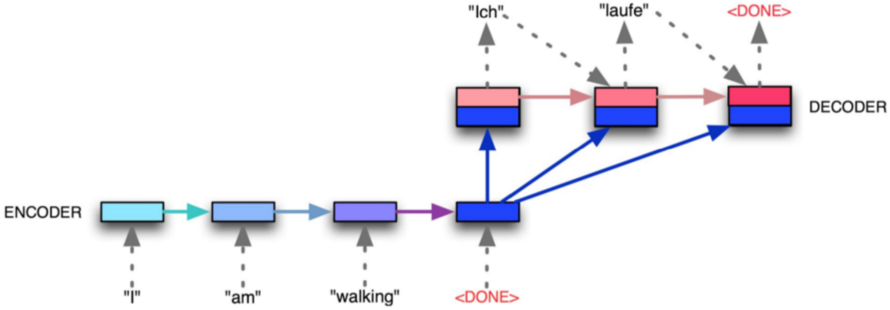
https://www.tensorflow.org/tutorials/seq2seq



Neural machine translation

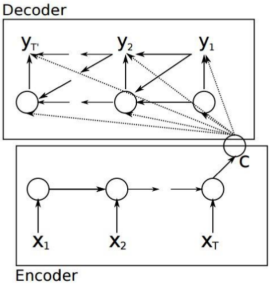
Stanford CS20si Lecture 13

Compare with in the simplest model depicted in last slide. It is more common to propagate the summary context **c** to all decoding iterations



Sequence-to-sequence model

Choi et al. 2014 (Bengio's group)



Consists of two RNNs:

- Encoder:

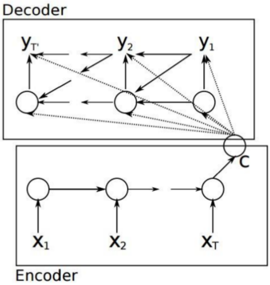
$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, x_t)$$
$$\mathbf{c} = \mathbf{h}_T$$

- Decoder:

$$\mathbf{s}_t = f(\mathbf{s}_{t-1}, y_{t-1}, \mathbf{c})$$
$$p(y_t|y_{t-1}, \cdots, y_1, \mathbf{x}) = g(\mathbf{s}_T, y_{t-1}, \mathbf{c})$$

N.B. $f(\cdot)$ at encoder and decoder are different

Sequence-to-sequence model
Choi et al. 2014 (Bengio's group)



Consists of two RNNs:

- Encoder:

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, x_t)$$
$$\mathbf{c} = \mathbf{h}_T$$

- Decoder:

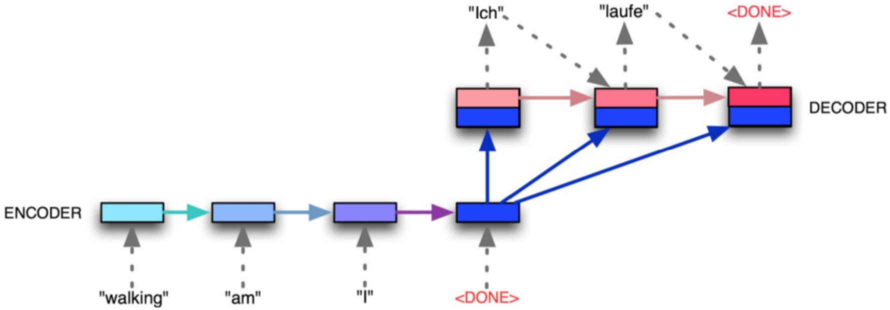
$$\mathbf{s}_t = f(\mathbf{s}_{t-1}, y_{t-1}, \mathbf{c})$$
$$p(y_t | y_{t-1}, \cdots, y_1, \mathbf{x}) = g(\mathbf{s}_T, y_{t-1}, \mathbf{c})$$

N.B. $f(\cdot)$ at encoder and decoder are different

Modification 1: Reverse input

Sutskever et al. 2014 (Hinton's group)

Reversing input was shown to improve results for NMT



Modification 2: Bucketing and padding

- When translating English to French, we expect English sentences of different lengths on input, and French sentences of different lengths on output
  - It will be infeasible to consider all different length combinations
- Instead, one may always consider a sufficiently large length and apply padding
  - Too much padding that leads to extraneous computation
- Bucketing is a method to efficiently handle sentences of different lengths
  - Group sequences of similar lengths into the same buckets
  - Create a separate subgraph for each bucket
  - E.g., translating ”I go” to french with a (4,6) bucket
    - Encoder input: [PAD ”.” ”go” I”]
    - Decoder input: [GO ”Je ”vais” ”.” EOS PAD]
    - Here we follow the tensorflow model that a special GO symbol is prepended to the decoder input

Modification 2: Bucketing and padding

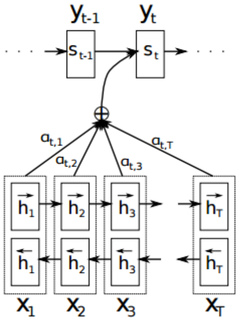
- When translating English to French, we expect English sentences of different lengths on input, and French sentences of different lengths on output
  - It will be infeasible to consider all different length combinations
- Instead, one may always consider a sufficiently large length and apply padding
  - Too much padding that leads to extraneous computation
- Bucketing is a method to efficiently handle sentences of different lengths
  - Group sequences of similar lengths into the same buckets
  - Create a separate subgraph for each bucket
  - E.g., translating "I go" to french with a (4,6) bucket
    - Encoder input: [PAD "." "go" I"]
    - Decoder input: [GO "Je "vais" "." EOS PAD]
    - Here we follow the tensorflow model that a special GO symbol is prepended to the decoder input

Modification 2: Bucketing and padding

- When translating English to French, we expect English sentences of different lengths on input, and French sentences of different lengths on output
  - It will be infeasible to consider all different length combinations
- Instead, one may always consider a sufficiently large length and apply padding
  - Too much padding that leads to extraneous computation
- Bucketing is a method to efficiently handle sentences of different lengths
  - Group sequences of similar lengths into the same buckets
  - Create a separate subgraph for each bucket
  - E.g., translating "I go" to french with a (4,6) bucket
    - Encoder input: [PAD "." "go" I"]
    - Decoder input: [GO "Je "vais" "." EOS PAD]
    - Here we follow the tensorflow model that a special GO symbol is prepended to the decoder input

Modification 3: Attention mechanism

Bahdanau et al. 2014 (Bengio's group)



- The original model summarizes the input with a single vector **c**
- Different output position probably more relevant to a part of the input

$$\mathbf{s}_t = f(\mathbf{s}_{t-1}, y_{t-1}, \mathbf{c}_t)$$
$$p(y_t | y_{t-1}, \cdots, y_1, \mathbf{x}) = g(\mathbf{s}_T, y_{t-1}, \mathbf{c}_t)$$

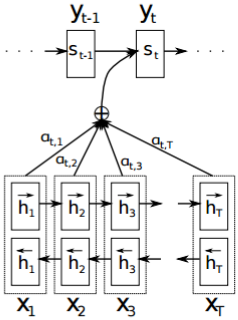
with

$$\mathbf{c}_i = \sum_j \alpha_{ij} \mathbf{h}_j, \qquad \alpha_{ij} = \frac{exp(e_{ij})}{\sum_k exp(e_{ik})}$$

where $e_{ij} = a(\mathbf{s}_{i-1}, \mathbf{h}_j)$ is an alignment score to see how well the inputs around position j matches output at position i

Modification 3: Attention mechanism

Bahdanau et al. 2014 (Bengio's group)



- The original model summarizes the input with a single vector **c**
- Different output position probably more relevant to a part of the input

$$\mathbf{s}_t = f(\mathbf{s}_{t-1}, y_{t-1}, \mathbf{c}_t)$$

$$p(y_t | y_{t-1}, \cdots, y_1, \mathbf{x}) = g(\mathbf{s}_T, y_{t-1}, \mathbf{c}_t)$$

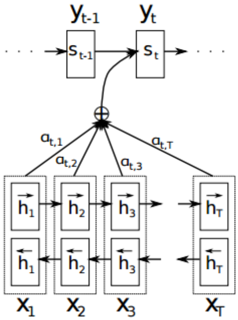
with

$$\mathbf{c}_i = \sum_j \alpha_{ij} \mathbf{h}_j, \qquad \alpha_{ij} = \frac{exp(e_{ij})}{\sum_k exp(e_{ik})}$$

where $e_{ij} = a(\mathbf{s}_{i-1}, \mathbf{h}_j)$ is an alignment score to see how well the inputs around position j matches output at position i

Modification 3: Attention mechanism

Bahdanau et al. 2014 (Bengio's group)



- The original model summarizes the input with a single vector **c**
- Different output position probably more relevant to a part of the input

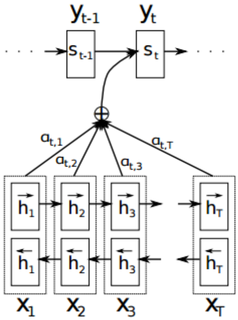
$$\mathbf{s}_t = f(\mathbf{s}_{t-1}, y_{t-1}, \mathbf{c_t})$$
$$p(y_t|y_{t-1}, \cdots, y_1, \mathbf{x}) = g(\mathbf{s}_T, y_{t-1}, \mathbf{c_t})$$

with

$$\mathbf{c}_i = \sum_j \alpha_{ij}\mathbf{h}_j, \qquad \alpha_{ij} = \frac{exp(e_{ij})}{\sum_k exp(e_{ik})}$$

where $e_{ij} = a(\mathbf{s}_{i-1}, \mathbf{h}_j)$ is an alignment score to see how well the inputs around position j matches output at position i

Modification 3: Attention mechanism
Bahdanau et al. 2014 (Bengio's group)



- The original model summarizes the input with a single vector **c**
- Different output position probably more relevant to a part of the input

$$\mathbf{s}_t = f(\mathbf{s}_{t-1}, y_{t-1}, \mathbf{c_t})$$
$$p(y_t|y_{t-1}, \cdots, y_1, \mathbf{x}) = g(\mathbf{s}_T, y_{t-1}, \mathbf{c_t})$$

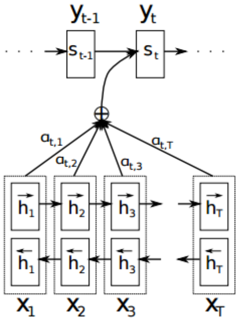
with

$$\mathbf{c}_i = \sum_j \alpha_{ij}\mathbf{h}_j, \qquad \alpha_{ij} = \frac{exp(e_{ij})}{\sum_k exp(e_{ik})}$$

where $e_{ij} = a(\mathbf{s}_{i-1}, \mathbf{h}_j)$ is an alignment score to see how well the inputs around position j matches output at position i

Modification 3: Attention mechanism

Bahdanau et al. 2014 (Bengio's group)



- The original model summarizes the input with a single vector $\mathbf{c}$
- Different output position probably more relevant to a part of the input

$$\mathbf{s}_t = f(\mathbf{s}_{t-1}, y_{t-1}, \mathbf{c_t})$$
$$p(y_t|y_{t-1}, \cdots, y_1, \mathbf{x}) = g(\mathbf{s}_T, y_{t-1}, \mathbf{c_t})$$

with

$$\mathbf{c}_i = \sum_j \alpha_{ij}\mathbf{h}_j, \qquad \alpha_{ij} = \frac{exp(e_{ij})}{\sum_k exp(e_{ik})}$$

where $e_{ij} = a(\mathbf{s}_{i-1}, \mathbf{h}_j)$ is an alignment score to see how well the inputs around position j matches output at position i

Modification 4: Softmax variations

- Recall the probability of getting each word w from softmax is like

$$p(w) = \frac{\exp(\mathcal{E}(w))}{\sum_w \exp(\mathcal{E}(w))}$$

- For a reasonable vocab size (say $\sim$ 50,000 words), the computation will be quite expensive
- Different approaches have been proposed in recent years to reduce the computation load. We will explain several here
  - Can check out this wonderful blog post by Sebastian Ruder if you are interested in this area

Modification 4: Softmax variations

- Recall the probability of getting each word w from softmax is like

$$p(w) = \frac{\exp(\mathcal{E}(w))}{\sum_w \exp(\mathcal{E}(w))}$$

- For a reasonable vocab size (say $\sim$ 50,000 words), the computation will be quite expensive
- Different approaches have been proposed in recent years to reduce the computation load. We will explain several here
  - Can check out this wonderful blog post by Sebastian Ruder if you are interested in this area

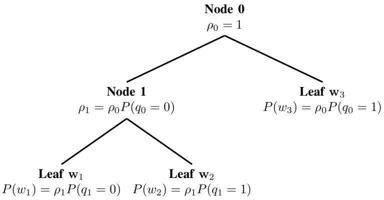
Modification 4: Softmax variations

- Recall the probability of getting each word w from softmax is like

$$p(w) = \frac{\exp(\mathcal{E}(w))}{\sum_w \exp(\mathcal{E}(w))}$$

- For a reasonable vocab size (say $\sim$ 50,000 words), the computation will be quite expensive
- Different approaches have been proposed in recent years to reduce the computation load. We will explain several here
  - Can check out this wonderful blog post by Sebastian Ruder if you are interested in this area

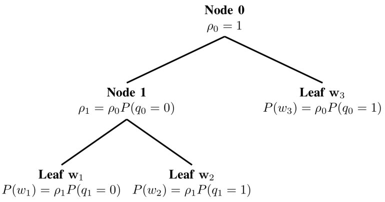
Modification 4a: Hierarchical softmax
Morin and Bengio 2005



- H-Softmax replaces the flat softmax layer with a hierarchical layer that has the words as leaves
- Decompose calculating the probability of one word into a sequence of probability calculations,
  - Saves us from having to calculate the expensive normalization over all words
- Replacing softmax with H-Softmax yields speedups of at least 50×
  - critical for low-latency tasks and used in Google's new messenger app Allo (yet another IM)

Modification 4a: Hierarchical softmax

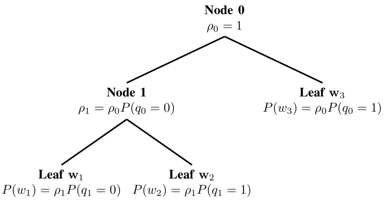
Morin and Bengio 2005



- H-Softmax replaces the flat softmax layer with a hierarchical layer that has the words as leaves
- Decompose calculating the probability of one word into a sequence of probability calculations,
  - Saves us from having to calculate the expensive normalization over all words
- Replacing softmax with H-Softmax yields speedups of at least 50×
  - critical for low-latency tasks and used in Google's new messenger app Allo (yet another IM)

Modification 4a: Hierarchical softmax

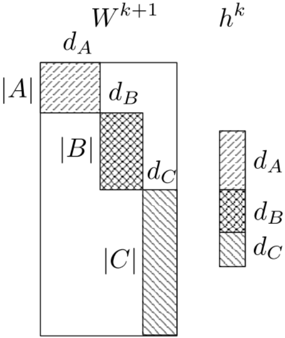
Morin and Bengio 2005



- H-Softmax replaces the flat softmax layer with a hierarchical layer that has the words as leaves
- Decompose calculating the probability of one word into a sequence of probability calculations,
  - Saves us from having to calculate the expensive normalization over all words
- Replacing softmax with H-Softmax yields speedups of at least 50×
  - critical for low-latency tasks and used in Google's new messenger app Allo (yet another IM)

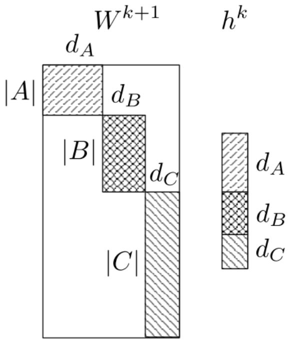
Modification 4b: Differentiated softmax

Chen et al. 2015



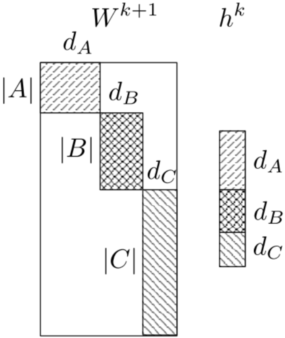
- Differentiated Softmax (D-Softmax) is based on the intuition that not all words require the same number of parameters
  - Many occurrences of frequent words allow us to fit many parameters to them
  - Extremely rare words might only allow to fit a few
- Instead of the dense matrix of the regular softmax layer of size $d \times |V|$
  - Embedding sizes increase with the frequencies of occurrence
- As many words will only require comparatively few parameters, the complexity of computing the softmax is reduced

Modification 4b: Differentiated softmax
Chen et al. 2015



- Differentiated Softmax (D-Softmax) is based on the intuition that not all words require the same number of parameters
  - Many occurrences of frequent words allow us to fit many parameters to them
  - Extremely rare words might only allow to fit a few
- Instead of the dense matrix of the regular softmax layer of size $d \times |V|$
  - Embedding sizes increase with the frequencies of occurrence
- As many words will only require comparatively few parameters, the complexity of computing the softmax is reduced

Modification 4b: Differentiated softmax
Chen et al. 2015



- Differentiated Softmax (D-Softmax) is based on the intuition that not all words require the same number of parameters
  - Many occurrences of frequent words allow us to fit many parameters to them
  - Extremely rare words might only allow to fit a few
- Instead of the dense matrix of the regular softmax layer of size $d \times |V|$
  - Embedding sizes increase with the frequencies of occurrence
- As many words will only require comparatively few parameters, the complexity of computing the softmax is reduced

Modification 4c: Sampled softmax

Jean et al. 2014 (Bengio's group)

- Consider a word $\tilde{w}$ and we try to find model parameter θ to maximize its log-probability

$$J_\theta = \log p(\tilde{w}) = \mathcal{E}(\tilde{w}) - \log \sum_w \exp \mathcal{E}(w)$$

Taking gradient w.r.t. θ, we have

$$\begin{aligned}\nabla_\theta J_\theta =& \nabla_\theta \mathcal{E}(\tilde{w}) - \sum_w \nabla_\theta \mathcal{E}(w) \frac{\exp(\mathcal{E}(w))}{\sum_w \exp(\mathcal{E}(w))} \\ =& \nabla_\theta \mathcal{E}(\tilde{w}) - \sum_w \nabla_\theta \mathcal{E}(w) p(w) = \nabla_\theta \mathcal{E}(\tilde{w}) - E\left[\nabla_\theta \mathcal{E}(w)\right],\end{aligned}$$

where $E\left[\nabla_\theta \mathcal{E}(w)\right]$ can be approximated with sampling

Modification 4c: Sampled softmax

Jean et al. 2014 (Bengio's group)

- Consider a word $\tilde{w}$ and we try to find model parameter θ to maximize its log-probability

$$J_\theta = \log p(\tilde{w}) = \mathcal{E}(\tilde{w}) - \log \sum_w \exp \mathcal{E}(w)$$

Taking gradient w.r.t. θ, we have

$$\begin{aligned}\nabla_\theta J_\theta =& \nabla_\theta \mathcal{E}(\tilde{w}) - \sum_w \nabla_\theta \mathcal{E}(w) \frac{\exp(\mathcal{E}(w))}{\sum_w \exp(\mathcal{E}(w))} \\ =& \nabla_\theta \mathcal{E}(\tilde{w}) - \sum_w \nabla_\theta \mathcal{E}(w) p(w) = \nabla_\theta \mathcal{E}(\tilde{w}) - E\left[\nabla_\theta \mathcal{E}(w)\right],\end{aligned}$$

where $E\left[\nabla_\theta \mathcal{E}(w)\right]$ can be approximated with sampling

Modification 4c: Sampled softmax

Jean et al. 2014 (Bengio's group)

- Consider a word $\tilde{w}$ and we try to find model parameter θ to maximize its log-probability

$$J_\theta = \log p(\tilde{w}) = \mathcal{E}(\tilde{w}) - \log \sum_w \exp \mathcal{E}(w)$$

Taking gradient w.r.t. θ, we have

$$\begin{aligned}\nabla_\theta J_\theta =& \nabla_\theta \mathcal{E}(\tilde{w}) - \sum_w \nabla_\theta \mathcal{E}(w) \frac{\exp(\mathcal{E}(w))}{\sum_w \exp(\mathcal{E}(w))} \\ =& \nabla_\theta \mathcal{E}(\tilde{w}) - \sum_w \nabla_\theta \mathcal{E}(w) p(w) = \nabla_\theta \mathcal{E}(\tilde{w}) - E\left[\nabla_\theta \mathcal{E}(w)\right],\end{aligned}$$

where $E\left[\nabla_\theta \mathcal{E}(w)\right]$ can be approximated with sampling

Modification 4c: Sampled softmax

Jean et al. 2014 (Bengio's group)

- Consider a word $\tilde{w}$ and we try to find model parameter θ to maximize its log-probability

$$J_\theta = \log p(\tilde{w}) = \mathcal{E}(\tilde{w}) - \log \sum_w \exp \mathcal{E}(w)$$

Taking gradient w.r.t. θ, we have

$$\begin{aligned}\nabla_\theta J_\theta =& \nabla_\theta \mathcal{E}(\tilde{w}) - \sum_w \nabla_\theta \mathcal{E}(w) \frac{\exp(\mathcal{E}(w))}{\sum_w \exp(\mathcal{E}(w))} \\ =& \nabla_\theta \mathcal{E}(\tilde{w}) - \sum_w \nabla_\theta \mathcal{E}(w) p(w) = \nabla_\theta \mathcal{E}(\tilde{w}) - E\left[\nabla_\theta \mathcal{E}(w)\right],\end{aligned}$$

where $E\left[\nabla_\theta \mathcal{E}(w)\right]$ can be approximated with sampling

Modification 4c: Sampled softmax

Jean et al. 2014 (Bengio's group)

- If we sample m number of w: $w_1, \cdots, w_m$, according to its distribution, we could approximate $E[\nabla_\theta \mathcal{E}(w)]$ as $\frac{1}{m}\sum_{i=1}^m \nabla_\theta \mathcal{E}(w_i)$
- Problem is that it is typically difficult to sample w precisely
- Instead, we can pick any distribution $q(w)$, and approximate $E[\nabla_\theta \mathcal{E}(w)]$ as a weighted sum

$$E\left[\nabla_\theta \mathcal{E}(w)\right] \approx \frac{1}{R}\sum_{i=1}^{m} r(w_i)\nabla_\theta \mathcal{E}(w_i),$$

  where $r(w) = \frac{\exp(\mathcal{E}(w))}{q(w)}$ corrects the discrepancy due to sampling from the "incorrect" distribution and $R = \sum_{i=1}^m r(w_i)$ is just a normalization factor
- The above is commonly known as **importance sampling** in statistics and $q(w)$ is known to be a **proposal distribution**

Modification 4c: Sampled softmax
Jean et al. 2014 (Bengio's group)

- If we sample m number of w: $w_1, \cdots, w_m$, according to its distribution, we could approximate $E[\nabla_\theta \mathcal{E}(w)]$ as $\frac{1}{m}\sum_{i=1}^m \nabla_\theta \mathcal{E}(w_i)$
- Problem is that it is typically difficult to sample w precisely
- Instead, we can pick any distribution $q(w)$, and approximate $E[\nabla_\theta \mathcal{E}(w)]$ as a weighted sum

$$E\left[\nabla_\theta \mathcal{E}(w)\right] \approx \frac{1}{R}\sum_{i=1}^{m} r(w_i)\nabla_\theta \mathcal{E}(w_i),$$

  where $r(w) = \frac{\exp(\mathcal{E}(w))}{q(w)}$ corrects the discrepancy due to sampling from the "incorrect" distribution and $R = \sum_{i=1}^m r(w_i)$ is just a normalization factor
- The above is commonly known as **importance sampling** in statistics and $q(w)$ is known to be a **proposal distribution**

Modification 4c: Sampled softmax

Jean et al. 2014 (Bengio's group)

- If we sample m number of w: $w_1, \cdots, w_m$, according to its distribution, we could approximate $E[\nabla_\theta \mathcal{E}(w)]$ as $\frac{1}{m}\sum_{i=1}^m \nabla_\theta \mathcal{E}(w_i)$
- Problem is that it is typically difficult to sample w precisely
- Instead, we can pick any distribution $q(w)$, and approximate $E[\nabla_\theta \mathcal{E}(w)]$ as a weighted sum

$$E\left[\nabla_\theta \mathcal{E}(w)\right] \approx \frac{1}{R}\sum_{i=1}^{m} r(w_i)\nabla_\theta \mathcal{E}(w_i),$$

where $r(w) = \frac{\exp(\mathcal{E}(w))}{q(w)}$ corrects the discrepancy due to sampling from the "incorrect" distribution and $R = \sum_{i=1}^m r(w_i)$ is just a normalization factor

- The above is commonly known as **importance sampling** in statistics and $q(w)$ is known to be a **proposal distribution**

Modification 4c: Sampled softmax

Jean et al. 2014 (Bengio's group)

- If we sample m number of w: $w_1, \cdots, w_m$, according to its distribution, we could approximate $E[\nabla_\theta \mathcal{E}(w)]$ as $\frac{1}{m}\sum_{i=1}^m \nabla_\theta \mathcal{E}(w_i)$
- Problem is that it is typically difficult to sample w precisely
- Instead, we can pick any distribution $q(w)$, and approximate $E[\nabla_\theta \mathcal{E}(w)]$ as a weighted sum

$$E\left[\nabla_\theta \mathcal{E}(w)\right] \approx \frac{1}{R}\sum_{i=1}^m r(w_i)\nabla_\theta \mathcal{E}(w_i),$$

  where $r(w) = \frac{\exp(\mathcal{E}(w))}{q(w)}$ corrects the discrepancy due to sampling from the "incorrect" distribution and $R = \sum_{i=1}^m r(w_i)$ is just a normalization factor
- The above is commonly known as **importance sampling** in statistics and $q(w)$ is known to be a **proposal distribution**

Modification 4c: Sampled softmax

Jean et al. 2014 (Bengio's group)

- If we sample m number of w: $w_1, \cdots, w_m$, according to its distribution, we could approximate $E[\nabla_\theta \mathcal{E}(w)]$ as $\frac{1}{m}\sum_{i=1}^m \nabla_\theta \mathcal{E}(w_i)$
- Problem is that it is typically difficult to sample w precisely
- Instead, we can pick any distribution $q(w)$, and approximate $E[\nabla_\theta \mathcal{E}(w)]$ as a weighted sum

$$E\left[\nabla_\theta \mathcal{E}(w)\right] \approx \frac{1}{R}\sum_{i=1}^m r(w_i)\nabla_\theta \mathcal{E}(w_i),$$

  where $r(w) = \frac{\exp(\mathcal{E}(w))}{q(w)}$ corrects the discrepancy due to sampling from the "incorrect" distribution and $R = \sum_{i=1}^m r(w_i)$ is just a normalization factor
- The above is commonly known as **importance sampling** in statistics and $q(w)$ is known to be a **proposal distribution**

Tensorflow demo

- You can find a tensorflow “demo” of its seq2seq model from one of its official tutorial
- The demo can translate English to French
- But you probably need a pretty big machine to try it. And it will consume 20G of space for data. Try it on schooner!

Chatbots
CS20si, assignment 3

Model based on the Google's vanilla translate model just described previously. Use features:

- Seq2seq
- Attentional decoder
- Reverse encoder inputs
- Bucketing
- Sampled softmax

Cornell Movie-Dialogs Corpus

Danescu-Niculescu-Mizil and Lee 2011

- 220,579 conversational exchanges between characters
- 10,292 pairs of movie characters
- 9,035 characters from 617 movies
- 304,713 total utterances
- Very well-formatted (almost perfect)

Come with a very interesting paper “Chameleons in Imagined Conversations”

Bucketing

- 9 buckets [(6, 8), (8, 10), (10, 12), (13, 15), (16, 19), (19, 22), (23, 26), (29, 32), (39, 44)] [19530, 17449, 17585, 23444, 22884, 16435, 17085, 18291, 18931]
- 5 buckets [(8, 10), (12, 14), (16, 19), (23, 26), (39, 43)] # bucket boundaries [37049, 33519, 30223, 33513, 37371] # number of samples in each bucket
- 3 buckets - **recommended** [37899, 34480, 31045] [(8, 10), (12, 14), (16, 19)] conversational exchanges between characters

The sassiest bot

```
> hi
you have you go .
> where?
in mind .
> you're weird?
you ' re right . . .
> what do you want?
who is you me .
> i don't understand
you know you look .
> i look what?
you nothing .
> you're funny
you ' re not .
```

> human
bot

The sassiest bot

```
> hi
hi .
> how're you?
just go .
> go where?
<unk> .
> omg
<unk> !
> you're doing that again!
i ' m not .
> stop telling me to go somewhere!
go !
> it's almost like you want to piss me off.
i ' m not .
> you're stupid
i ' m not .
```

> human
bot

It sort of works

```
> you are a monster!
i am . . .
> i'm sorry
you don ' t need to boy .
> i'm a girl!
you done ?
==========
> Did you change your hair?
no .
> no?
no i won t .
> who?
paul owen .
> where?
in a harem .
```

> human
bot

Sort of doesn't

```
> what?
don ' t bother .
> you're being sassy
i need .
> what do you need?
i need ya help ya .
> okay
what do you want ?
```

> human
bot

Sort of doesn't

```
> you're just talking gibberish
and what are you talking about ?
> about life
and what about it ?
> life is beautiful
is good .
> where do you live?
up .
> as in heaven?
in his apartment .
> who's he?
who ' s here ?
> oh well
and you are in trouble .
> i'm not!
please !
> whatever
i don ' t know what to say .
```

> human
bot

Demo

Problems?

- The bot is very dramatic (thanks to Hollywood screenwriters)
- Topics of conversations aren't realistic
- Responses are always the same given the same input
- Inconsistent personality
- Use only the last previous utterance as the input for the encoder
- Doesn't keep track of information about users

Potential improvement 1: Train on multiple datasets

- Twitter chat log (courtesy of Marsan Ma)
- Every publicly available Reddit comments (1TB of data!)
- Your own conversations (chat logs, text messages, emails)

Potential improvement 2: Add personalities?

- At the decoder phase, inject consistent information about the bot
  For example: name, age, hometown, current location, job
- Use the decoder inputs from one person only

Potential improvement 3: Train on the incoming inputs

- Save the conversation with users and train on those conversations
- Create a feedback loop so users can correct the bot's responses

Potential improvement 4: Remember what users say

- The bot can extract information the user gives them

```
> hi
hi . what ' s your name ?
> my name is chip
nice to meet you .
> what's my name?
let ' s talk about something else .
```

Conclusions

- Sequence-to-sequence models are the state-of-the-art for machine translation
- But seq2seq are not just for language translation. Potentially it can "translate" any 1-D signal to one form to another

Conclusions

- Sequence-to-sequence models are the state-of-the-art for machine translation
- But seq2seq are not just for language translation. Potentially it can "translate" any 1-D signal to one form to another