NEURAL MACHINE TRANSLATION (PART 2)

David Talbot

23rd April, 2017

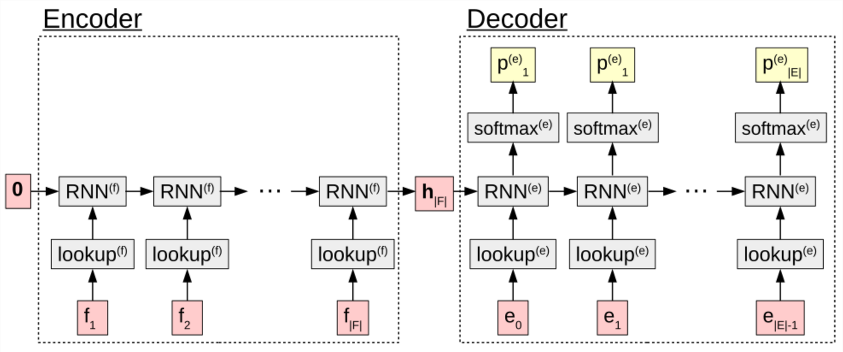
Computer Science Club, St. Petersburg, Russia

today's topics

- Encoder-decoder models (re-cap)
- Inductive bias in NMT
    - Attention
    - Coverage
    - Source-target context
- Softmax approximations
- Open vocabulary NMT
- Other topics
- References

ENCODER-DECODER MODELS (RE-CAP)

encoder decoder nmt (recap)



(From Graham Neubig, Draft book on NMT)

encoder decoder parameters

- $W^{(f)}$ source embedding matrix
- $W^{(e)}$ target embedding matrix
- $RNN^{(f)}$ source state update parameters (LSTM, GRU etc.)
- $RNN^{(e)}$ target state update parameters (LSTM, GRU etc.)
- $W^{(s)}, \mathbf{b}^{(s)}$ softmax output weight matrix and bias

encoder decoder nmt computations

Encode source sentence

- Initialize encoder state $\mathbf{h}_0^{(f)}$ to 0
- Look up embeddings for source words f_j for $j \in [1, J]$

$$\mathbf{w}_{f_j}^{(f)} = W_{\cdot, f_j}^{(f)}$$

- Compute encoder state using $RNN^{(f)}$

$$\mathbf{h}_j^{(f)} = RNN^{(f)}(\mathbf{w}_{f_j}, \mathbf{h}_{j-1}^{(f)})$$

bi-directional encoder (aside)

The encoder could process the input from both left-to-right and right-to-left giving

$$\overrightarrow{h}_j^{(f)} = \overrightarrow{RNN}^{(f)}(\mathrm{w}_{f_j}, \mathrm{h}_{j-1}^{(f)})$$

and

$$\overleftarrow{h}_j^{(f)} = \overleftarrow{RNN}^{(f)}(\mathrm{w}_{f_j}, \mathrm{h}_{j+1}^{(f)})$$

and state is a concatenation

$$\mathrm{h}_j^{(f)} = [\overrightarrow{\mathrm{h}}_j^{(f)}, \overleftarrow{\mathrm{h}}_j^{(f)}].$$

encoder decoder computations

Decode target sentence

- Initialize decoder state $\mathbf{h}_0^{(e)}$ to $\mathbf{h}_J^{(f)}$
- Compute distribution over target words using softmax

$$\mathbf{p}_i^{(e)} \propto \exp(W^{(s)}\mathbf{h}_i^{(e)} + \mathbf{b}^{(s)})$$

- Choose a target word e_i and look up its embedding

$$\mathbf{w}_{e_i}^{(e)} = W_{\cdot,e_i}^{(e)}$$

- Update decoder state with embedding of last target word

$$\mathbf{h}_i^{(e)} = RNN^{(e)}(\mathbf{w}_{e_{i-1}}^{(e)}, \mathbf{h}_{i-1}^{(e)})$$

- Stop when end of sentence symbol $< /s >$ is generated

How to choose each target word from decoder?

- A random sample: condition on current output choosing

$$e_i \sim \mathsf{Pr}(e_i | e_1, ..., e_{i-1}, \mathsf{f}),\ \textit{i.e.}\ \mathsf{p}_i^{(e)}.$$

- Greedy search: choose most likely target word at each step

$$e_i = \arg\max_{e \in V_e} \mathsf{Pr}(e_i | e_1, ..., e_{i-1}, \mathsf{f}),\ \textit{i.e.}\ \mathsf{p}_i^{(e)}$$

- Beam search?

beam search

- Start with n most probable hypotheses for e_1
- Maintain separate decoder state for each $\mathbf{h}^{(e)}_{i,k},\ k \in [1, n]$
- Expand the top n hypotheses with their top n continutations
- Prune expanded set back to top n hypotheses
- Store each hypothesis that emits $< /s >$ and set $n = n - 1$
- Output top scoring stored hypothesis once $n = 0$

beam search

- Start with n most probable hypotheses for e_1
- Maintain separate decoder state for each $\mathbf{h}_{i,k}^{(e)},\ k \in [1, n]$
- Expand the top n hypotheses with their top n continutations
- Prune expanded set back to top n hypotheses
- Store each hypothesis that emits $< /s >$ and set $n = n - 1$
- Output top scoring stored hypothesis once $n = 0$

How will increasing n affect target sentence length?

beam search

- Start with n most probable hypotheses for e_1
- Maintain separate decoder state for each $\mathbf{h}_{i,k}^{(e)},\ k \in [1, n]$
- Expand the top n hypotheses with their top n continutations
- Prune expanded set back to top n hypotheses
- Store each hypothesis that emits $< /s >$ and set $n = n - 1$
- Output top scoring stored hypothesis once $n = 0$

How will increasing n affect target sentence length?
How and why might we adapt the beam size n during decoding?

modelling target sentence length

- Intrinsic bias of product model for shorter outputs (?)

$$\mathrm{Pr}(\mathbf{e}|\mathbf{f}) = \prod_{i=1}^{I} \mathrm{Pr}(e_i|e_1, ..., e_{i-1}, \mathbf{f}).$$

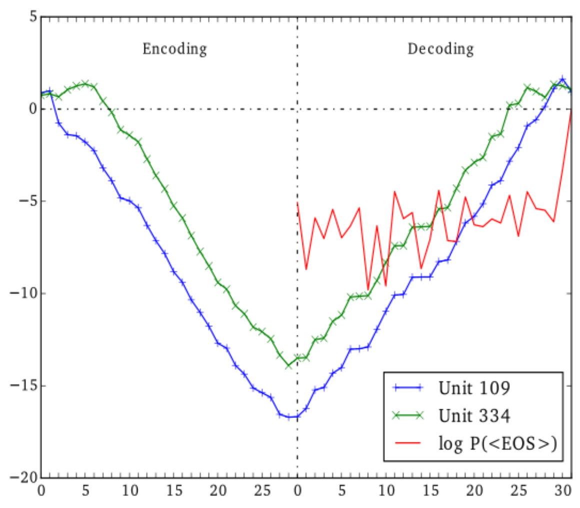
- Model the target sentence length I conditioned on the source sentence length J estimated from training corpus

$$\mathrm{Pr}(I = i|J = j) = \frac{\#(i, j)}{\#(j)}.$$

- Normalize the log probability by I and choose

$$\mathbf{e} = \operatorname*{argmax}_{\mathbf{e}} \log \frac{\mathrm{Pr}(\mathbf{e}|\mathbf{f})}{I}.$$

but maybe some of the rnn states already model length (?)



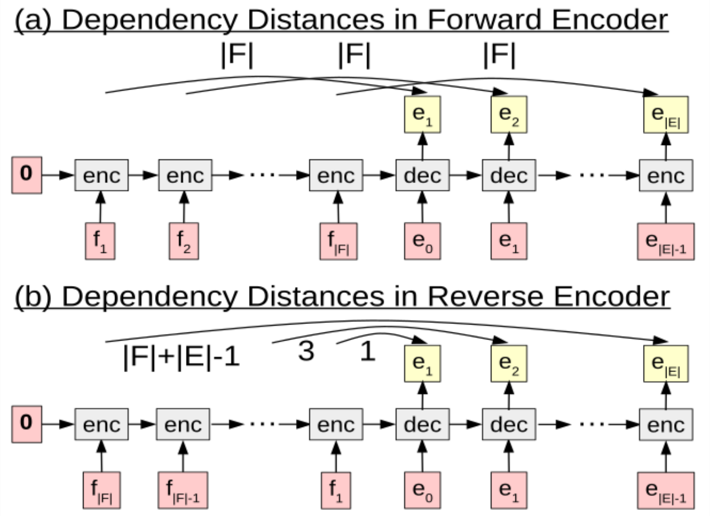
(From Shi et al. 2016)

INDUCTIVE BIAS IN NMT

problems with vanilla encoder-decoder nmt

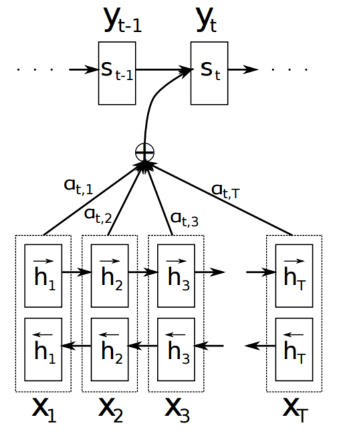
1. Long range dependencies between source and target
2. Fixed size memory: the final state of encoder ($\mathbf{h}_J^{(f)}$)
3. No MT specific *inductive bias*, e.g. coverage, reordering constraints, etc.

dependencies in vanilla nmt



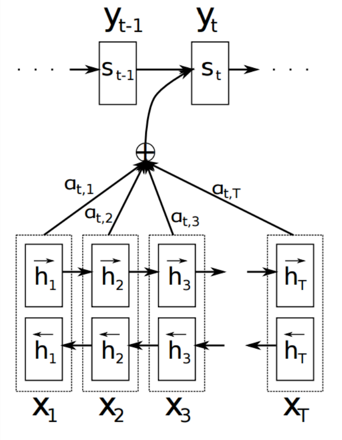
(From Neubig 2017)

attention (bahdanau et al. 2015)



Each target word conditioned on a different source context.

attention (bahdanau et al. 2015)



Decoder RNN transitions also take $\mathbf{c}_i$

$$\mathbf{h}_i^{(e)} = RNN^{(e)}(\mathbf{w}_{e_{i-1}}^{(e)}, \mathbf{h}_{i-1}^{(e)}, \mathbf{c}_i)$$

attention mechanism

- Given encoder matrix $H^{(f)}$ where column j is $\mathsf{h}_j^{(f)}$ for $j \in [1, J]$
- Compute the source context c_i for target word i as

$$\mathsf{c}_i = H^{(f)} \alpha_i$$

  where α_i is a positive vector that sums to 1.
- Element k of α_i indicates importance of source word k for generating hidden state for target word i.

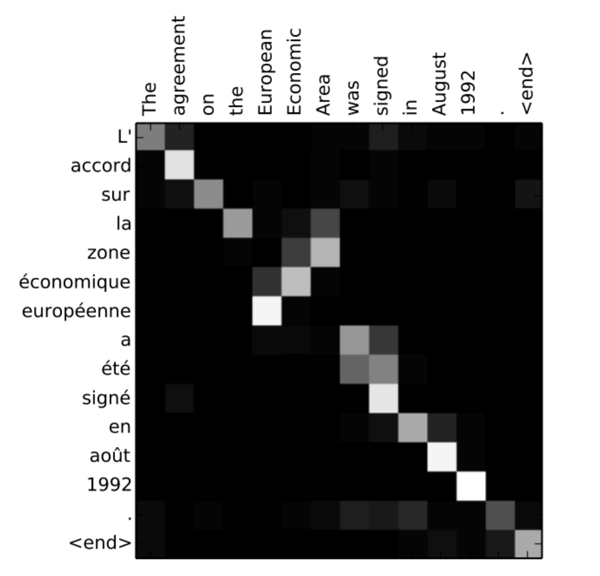
attention mechanism

- Given encoder matrix $H^{(f)}$ where column j is $\mathbf{h}_j^{(f)}$ for $j \in [1, J]$
- Compute the source context $\mathbf{c}_i$ for target word i as

$$\mathbf{c}_i = H^{(f)} \alpha_i$$

  where α_i is a positive vector that sums to 1.
- Element k of α_i indicates importance of source word k for generating hidden state for target word i.
- How do we compute α_i?

attention as alignment



attention mechanism computations

1. Initialize $\mathbf{c}_0 = \mathbf{0}$.
2. Update decoder state $\mathbf{h}_i^{(e)}$ as
$$\mathbf{h}_i^{(e)} = RNN^{(e)}(\mathbf{w}_{i-1}^{(e)}, \mathbf{h}_{i-1}^{(e)}, \mathbf{c}_{i-1}).$$
3. Compute unnormalized attention score $\mathbf{a}_{(i,j)}$ from these
$$\mathbf{a}_{(i,j)} = ATTENTION(\mathbf{h}_j^{(f)}, \mathbf{h}_i^{(e)}).$$
4. Use softmax to get α_i to be a probability distribution, i.e.
$$\alpha_{(i,j)} = \frac{\exp(\mathbf{a}_{(i,j)})}{\sum_k \exp(\mathbf{a}_{(i,k)})}.$$
5. Compute context $\mathbf{c}_i = H^{(f)}\alpha_i$
6. Use context $\mathbf{c}_i$ and decoder state $\mathbf{h}_i^{(e)}$ in output softmax.

attention models (neubig 2017)

- Dot product

$$ATTENTION(\mathbf{h}_j^{(f)}, \mathbf{h}_i^{(e)}) = \mathbf{h}_j^{(f)T}\mathbf{h}_i^{(e)}.$$

- Bilinear model

$$ATTENTION(\mathbf{h}_j^{(f)}, \mathbf{h}_i^{(e)}) = \mathbf{h}_j^{(f)T} W_a \mathbf{h}_i^{(e)}.$$

- Multi-layer perceptron (Bahdanau et al. 2015)

$$ATTENTION(\mathbf{h}_j^{(f)}, \mathbf{h}_i^{(e)}) = \mathbf{w}_a^T \tanh(W_a[\mathbf{h}_i^{(e)}; \mathbf{h}_j^{(f)}]).$$

attention: improves longer sentences (bahdanau et al. 2015)

- (source)
  An admitting privilege is the right of a doctor to admit a patient to a hospital <u>to carry out a diagnosis or a procedure based on his status as a health care worker at a hospital.</u>
- (NMT without attention)
  Un privilege d'admission est le droit d'un médecin de reconnaître un patient à l'hôpital <u>d'un diagnostic ou de prendre un diagnostic en fonction de son état de santé.</u>
- (NMT with attention)
  Un privilege d'admission est le droit d'un médecin d'admettre un patient à un hôpital
  <u>pour effectuer un diagnostic ou une procédure,</u>
  <u>selon son statut de travailleur des soins de santé à l'hopital.</u>

comparing nmt and pbmt (1)

- Phrase-based Machine Translation
  - Lots of weak features
  - Lots of independence assumptions
  - Model backs off to simpler features due to sparsity
  - Multiple steps to training pipeline: alignment, phrase extraction, model estimation, tuning etc.
- Neural Machine Translation
  - Trained end-to-end to optimize likelihood (or metric)
  - Continuous space alleviates sparsity
  - No explicit independence assumptions

comparing nmt and pbmt (2)

- NMT with attention is state-of-the-art (since late 2015)
- Human evaluations suggest NMT is more *fluent* than PBMT
- Automatic evaluations support this (Bentivogli 2016):
    - Better with inflections: when NMT chooses the correct *lemma*, it is more likely to choose the correct form.
    - Fewer reordering errors for NMT.

comparing nmt and pbmt (2)

- NMT with attention is state-of-the-art (since late 2015)
- Human evaluations suggest NMT is more *fluent* than PBMT
- Automatic evaluations support this (Bentivogli 2016):
  - Better with inflections: when NMT chooses the correct *lemma*, it is more likely to choose the correct form.
  - Fewer reordering errors for NMT.

But is lexical choice worse for NMT? (i.e. adequacy)

common nmt failures (1)

source:

Using this machine does not quite require a lot of skill .

target:

В этом случае не менее , чтобы быть не только много .

common nmt failures (1)

source:

Using this machine does not quite require a lot of skill .

target:

В этом случае не менее , чтобы быть не только много .

Fluency over adequacy (i.e. prefers to make target sound good and ignore the source).

common nmt failures (2)

source:

State institutions began playing a very active role in allocating property – both state-owned and private .

target:

В этой году , что в том , что в том , что в том , что в этом году и в области .

common nmt failures (2)

source:

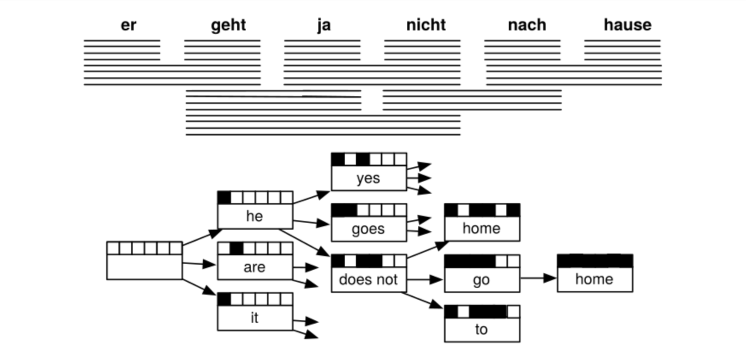
State institutions began playing a very active role in allocating property – both state-owned and private .

target:

В этой году , что в том , что в том , что в том , что в этом году и в области .

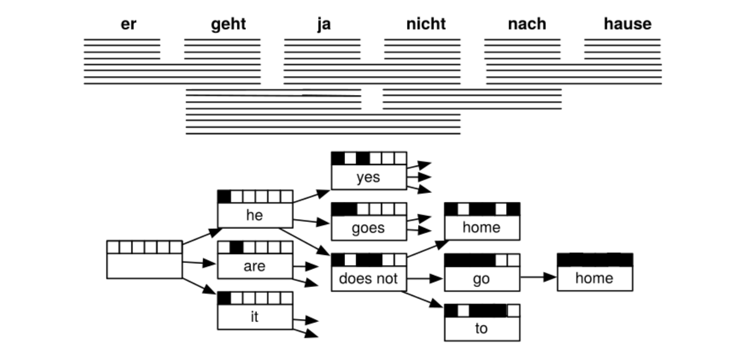
Repetition: over- and under- translation of source words.

coverage



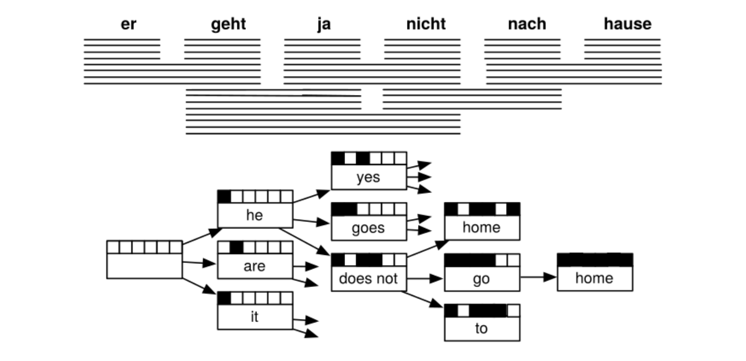
- PBMT tracks which words have been translated.

coverage



- PBMT tracks which words have been translated.
- Advanced IBM alignment models model *fertility* (i.e. how many target words to translate each source word into).

coverage



- PBMT tracks which words have been translated.
- Advanced IBM alignment models model *fertility* (i.e. how many target words to translate each source word into).
- Attention models (so far) don't explicitly track coverage.

coverage

- PBMT coverage is hard and can be n-to-m (due to phrases).
- NMT attention is soft and n-to-1 since words are generated one-by-one.

coverage (mi et al. 2016)

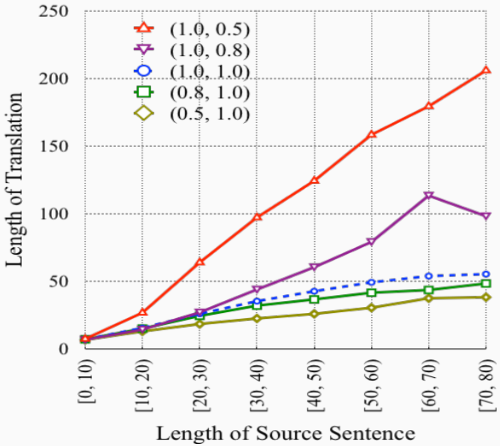
- Initialize coverage vectors $\mathbf{c}_{0,j}$ for $j \in [1, J]$ based on source words (cf. per source fertility parameter in IBM model 4).
- Update each vector based on attention and generated target using either GRU or just subtraction

$$\mathbf{c}_{i,f_j} = \mathbf{c}_{i-1,f_j} - \alpha_{i,j} \circ (W^{(e \to c)} e_i)$$

  where $W^{(e \to c)}$ is a matrix that converts target output embeddings to coverage embeddings space.
- Feed coverage vectors to attention model (MLP).

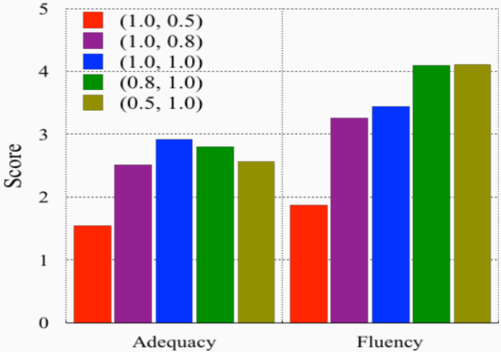
Also try explicitly minimizing coverage vector norms from reference alignments (i.e. force them to zero).

source vs. target context gating (tu et al. 2016)



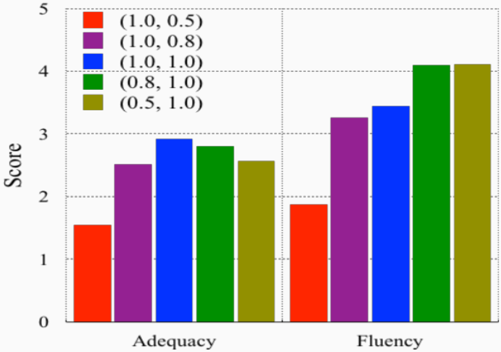
Reducing source context, increases target length.

source vs. target context gating (tu et al. 2016)



Reducing source context, increases fluency.

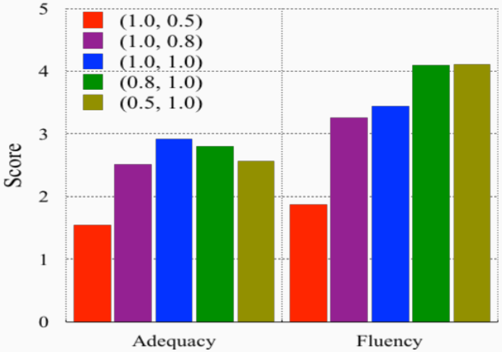
source vs. target context gating (tu et al. 2016)



Reducing source context, increases fluency.

Reducing target context, doesn't increase adequacy:

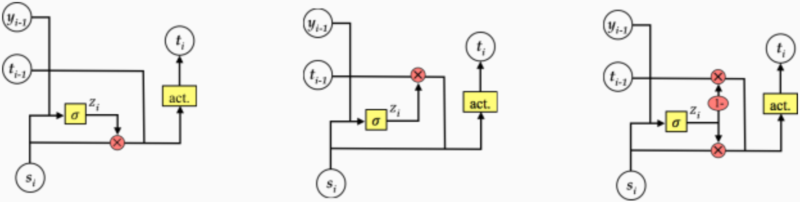
source vs. target context gating (tu et al. 2016)



Reducing source context, increases fluency.

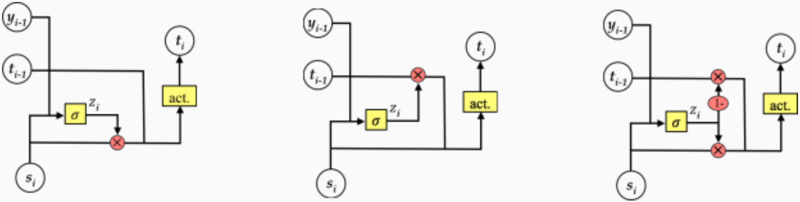
Reducing target context, doesn't increase adequacy: it just results in repetitions.

source-target context gates (tu et al. 2016)



Source and target context gates similar to LSTM output gate

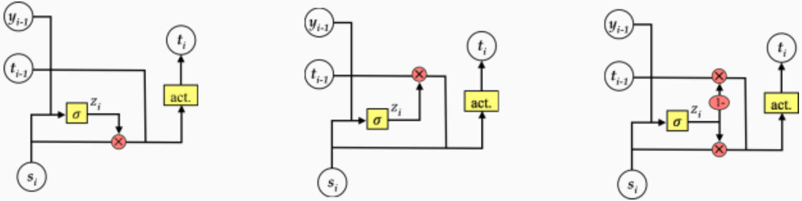
source-target context gates (tu et al. 2016)



Source and target context gates similar to LSTM output gate

Gate for both similar to update gate in GRU

source-target context gates (tu et al. 2016)



Source and target context gates similar to LSTM output gate

Gate for both similar to update gate in GRU

(Tu et al. 2016) use context gate instead of GRU with plain RNN.

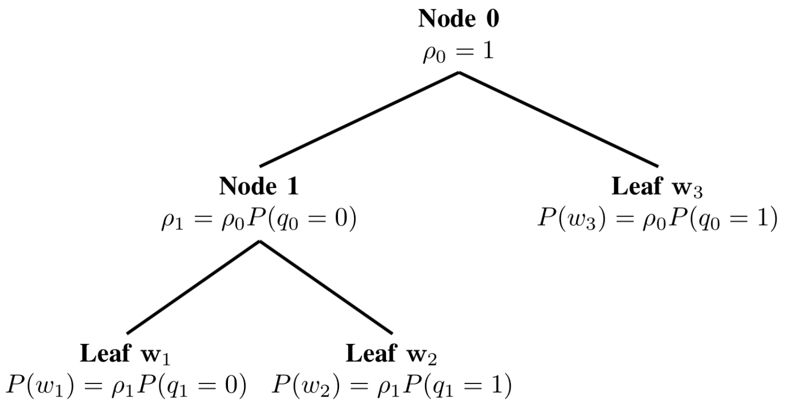
SOFTMAX APPROXIMATIONS

softmax is a computational bottleneck

Compares output of penultimate layer $\mathbf{h}$ with each target word's embedding

$$\Pr(w|\mathbf{h}) = \frac{\exp(\mathbf{h}^T \mathbf{v}_w)}{\sum_{w' \in \mathcal{V}} \exp(\mathbf{h}^T \mathbf{v}_w)}$$

hierarchical softmax (morin & bengio 2005)



Assign each target word a unique binary code

hierarchical softmax (morin & bengio 2005)

Given the path $L(w)$ from root to word w compute

$$\Pr(w|\mathbf{h}) = \prod_{d_i \in L(w)} \Pr(d_i|\mathbf{q}_i, \mathbf{h}).$$

where $d_i \in \{0, 1\}$ is the i-th digit of path and $\mathbf{q}_i$ is the feature vector at i-th node. The probability at each node is given by

$$\Pr(d_i = 1|\mathbf{q}_i, \mathbf{h}) = \sigma(\mathbf{h}^T\mathbf{q}_i + b_i).$$

hierarchical softmax (morin & bengio 2005)

Given the path $L(w)$ from root to word w compute

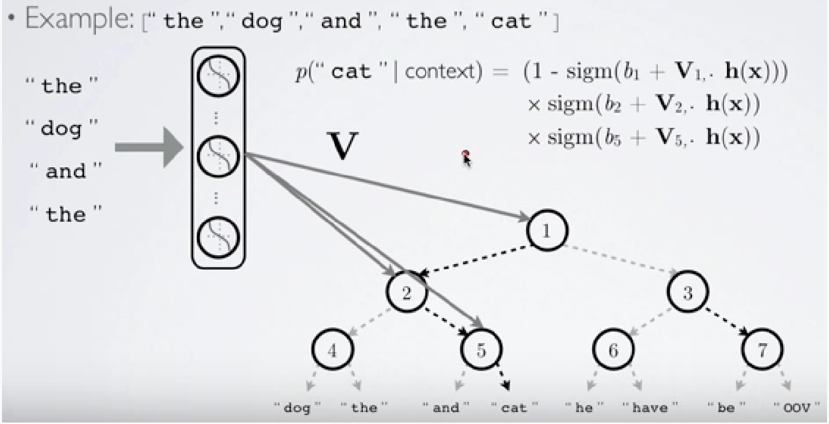
$$\Pr(w|\mathbf{h}) = \prod_{d_i \in L(w)} \Pr(d_i|\mathbf{q}_i, \mathbf{h}).$$

where $d_i \in \{0, 1\}$ is the i-th digit of path and $\mathbf{q}_i$ is the feature vector at i-th node. The probability at each node is given by

$$\Pr(d_i = 1|\mathbf{q}_i, \mathbf{h}) = \sigma(\mathbf{h}^T \mathbf{q}_i + b_i).$$

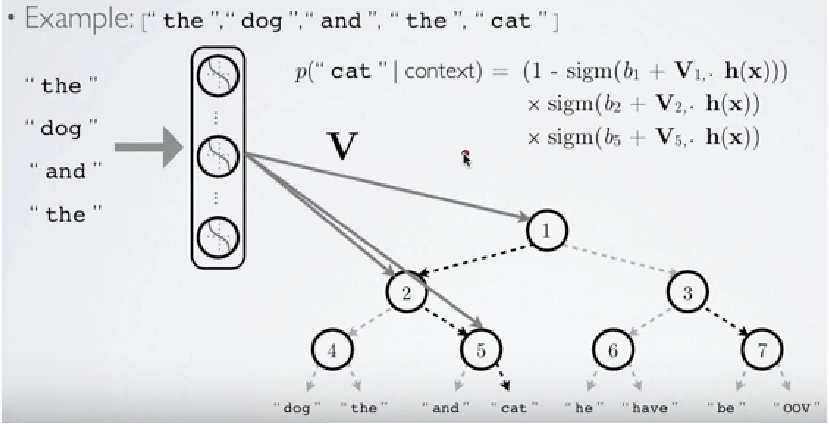
There are $\log_2(|\mathcal{V}|)$ nodes on path $L(w)$ and $|\mathcal{V}|$ nodes (i.e. output parameter vectors) in total.

hierarchical softmax (morin & bengio 2005)



During training, only update parameters on the correct path.

hierarchical softmax (morin & bengio 2005)



During training, only update parameters on the correct path.

At test time we don't get these speed-ups.

hierarchical softmax clustering

How to partition $\mathcal{V}$ into learnable classes?

- WordNet clusters (Morin & Bengio 2005) performs poorly
- Data driven clustering: e.g. build a random binary tree, learn embedding, then cluster words by their embeddings using binary mixture model at each node (Mhin & Hinton 2009)
- Huffman tree (Mikolov et. al 2013)

softmax with cross-entropy loss

Using a softmax output layer the cross-entropy loss becomes

$$C_{cross\_entropy} \equiv -\frac{1}{n}\sum_{i=1}^{n} \log \hat{\Pr}(w_n)$$

softmax with cross-entropy loss

Using a softmax output layer the cross-entropy loss becomes

$$\begin{aligned} C_{cross\_entropy} &\equiv -\frac{1}{n}\sum_{i=1}^{n} \log \hat{\Pr}(w_n) \\ &= -\frac{1}{n}\sum_{i=1}^{n} \log \frac{\exp(\mathbf{h}^T \mathbf{v}_{w_n})}{\sum_{w' \in \mathcal{V}} \exp(\mathbf{h}^T \mathbf{v}'_w)} \end{aligned}$$

softmax with cross-entropy loss

Using a softmax output layer the cross-entropy loss becomes

$$
\begin{aligned}
C_{cross\_entropy} &\equiv -\frac{1}{n}\sum_{i=1}^{n} \log \hat{\mathbf{Pr}}(w_n) \\
&= -\frac{1}{n}\sum_{i=1}^{n} \log \frac{\exp(\mathbf{h}^T \mathbf{v}_{w_n})}{\sum_{w' \in \mathcal{V}} \exp(\mathbf{h}^T \mathbf{v}'_w)} \\
&= -\frac{1}{n}\sum_{i=1}^{n} \left(\mathbf{h}^T \mathbf{v}_{w_n} - \log \sum_{w' \in \mathcal{V}} \exp(\mathbf{h}^T \mathbf{v}'_w) \right).
\end{aligned}
$$

gradient of softmax with cross-entropy loss

The gradient involves a positive reinforcement term for the correct word and a penalty for all words.

$$\nabla_\theta C_{cross\_entropy} \quad = \quad -\nabla_\theta \mathbf{h}^T v_{w_n} + \nabla_\theta \log \sum_{w' \in \mathcal{V}} \exp(\mathbf{h}^T \mathbf{v}'_w)$$

gradient of softmax with cross-entropy loss

The gradient involves a positive reinforcement term for the correct word and a penalty for all words.

$$
\begin{aligned}
\nabla_\theta C_{cross\_entropy} &= -\nabla_\theta \mathbf{h}^T v_{w_n} + \nabla_\theta \log \sum_{w' \in \mathcal{V}} \exp(\mathbf{h}^T \mathbf{v}'_w) \\
&= -\nabla_\theta \mathbf{h}^T v_{w_n} + \sum_{w' \in \mathcal{V}} \frac{\exp(\mathbf{h}^T \mathbf{v}_{w'})}{\sum_{\hat{w} \in \mathcal{V}} \exp(\mathbf{h}^T \mathbf{v}_{\hat{w}})} \nabla_\theta \mathbf{h}^T \mathbf{v}'_w
\end{aligned}
$$

gradient of softmax with cross-entropy loss

The gradient involves a positive reinforcement term for the correct word and a penalty for all words.

$$\begin{aligned}
\nabla_\theta C_{cross_entropy} &= -\nabla_\theta \mathbf{h}^T v_{w_n} + \nabla_\theta \log \sum_{w' \in \mathcal{V}} \exp(\mathbf{h}^T \mathbf{v}'_w) \\
&= -\nabla_\theta \mathbf{h}^T v_{w_n} + \sum_{w' \in \mathcal{V}} \frac{\exp(\mathbf{h}^T \mathbf{v}_{w'})}{\sum_{\hat{w} \in \mathcal{V}} \exp(\mathbf{h}^T \mathbf{v}_{\hat{w}})} \nabla_\theta \mathbf{h}^T \mathbf{v}'_w \\
&= -\nabla_\theta \mathbf{h}^T v_{w_n} + \sum_{w' \in \mathcal{V}} \hat{\Pr}(w') \nabla_\theta \mathbf{h}^T \mathbf{v}'_w
\end{aligned}$$

gradient of softmax with cross-entropy loss

The gradient involves a positive reinforcement term for the correct word and a penalty for all words.

$$
\begin{aligned}
\nabla_\theta C_{cross\_entropy} &= -\nabla_\theta \mathbf{h}^T v_{w_n} + \nabla_\theta \log \sum_{w' \in \mathcal{V}} \exp(\mathbf{h}^T \mathbf{v}'_w) \\
&= -\nabla_\theta \mathbf{h}^T v_{w_n} + \sum_{w' \in \mathcal{V}} \frac{\exp(\mathbf{h}^T \mathbf{v}_{w'})}{\sum_{\hat{w} \in \mathcal{V}} \exp(\mathbf{h}^T \mathbf{v}_{\hat{w}})} \nabla_\theta \mathbf{h}^T \mathbf{v}'_w \\
&= -\nabla_\theta \mathbf{h}^T v_{w_n} + \sum_{w' \in \mathcal{V}} \hat{\mathrm{Pr}}(w') \nabla_\theta \mathbf{h}^T \mathbf{v}'_w \\
&= -\nabla_\theta \mathbf{h}^T v_{w_n} + \mathbb{E}_{\hat{\mathrm{Pr}}}[\nabla_\theta \mathbf{h}^T \mathbf{v}'_w].
\end{aligned}
$$

where $\hat{\mathrm{Pr}}$ is the softmax distribution and $\mathbb{E}$ is the expected value under this distribution.

Basic Monte-Carlo approximates expected value of $f(X)$ under distribution P as a *stochastic* average

$$\mathbb{E}_P[f(X)] \approx \frac{1}{n}\sum_{i=1}^{n} f(X_i)$$

where $X \sim P$ and $X_i \sim P$.

importance sampling

If it's hard to sample from P, use another distribution Q. Then

$$\mathbb{E}_P[f(X)] \approx \frac{1}{n}\sum_{i=1}^{n} f(X_i)\omega(X_i)$$

where $X_i \sim Q$, $P(X) > 0 \Rightarrow Q(X) > 0$ and $\omega(X)$ are *importance weights*. If $\omega(X) = \frac{P(X)}{Q(X)}$ this is an unbiased estimator of $f(X)$.

Choosing a good Q can reduce variance over basic Monte Carlo.

vocabulary subset importance sampling (jean et al. 2015)

Biased importance sampler (seen elsewhere in NLP)

- Partition training corpus and let V_i be vocabulary for i-th part
- Define proposal distribution Q_i for i-th section as

$$Q_i(w) = \begin{cases} \frac{1}{|V_i|}, & \text{if } w \in V_i \\ 0, & \text{otherwise} \end{cases}$$

Define importance weights as

$$\omega_i(w) = \frac{\omega_i'(w)}{\sum_{w' \in V_i} \omega_i'(w')}$$

where

$$\omega_i'(w) = \frac{\exp(\mathbf{h}^T \mathbf{v}_w)}{Q_i(w)}.$$

vocabulary subset importance sampling (jean et al. 2015)

Biased importance sampler (seen elsewhere in NLP)

- Partition training corpus and let V_i be vocabulary for i-th part
- Define proposal distribution Q_i for i-th section as

$$Q_i(w) = \begin{cases} \frac{1}{|V_i|}, & \text{if } w \in V_i \\ 0, & \text{otherwise} \end{cases}$$

Define importance weights as

$$\omega_i(w) = \frac{\omega_i'(w)}{\sum_{w' \in V_i} \omega_i'(w')}$$

where

$$\omega_i'(w) = \frac{\exp(\mathbf{h}^T \mathbf{v}_w)}{Q_i(w)}.$$

Allows us to choose $|V_i|$ based on GPU memory ...

adaptive importance sampling (jean et al. 2015)

Since Q_i is uniform the terms cancel so the approximation becomes

$$\begin{aligned}
\nabla_\theta C_{cross_entropy} &= -\nabla_\theta \mathbf{h}^T v_{w_n} + \mathbb{E}_{\hat{\mathbf{Pr}}}[\nabla_\theta \mathbf{h}^T w'] \\
&\approx -\nabla_\theta \mathbf{h}^T v_{w_n} + \sum_{w' \in V_i} \frac{\omega_i'(w')}{\sum_{\hat{w} \in V_i} \omega_i'(\hat{w})} \nabla_\theta \mathbf{h}^T \mathbf{v}_w' \\
&= -\nabla_\theta \mathbf{h}^T v_{w_n} + \sum_{w' \in V_i} \frac{\exp(\mathbf{h}^T \mathbf{v}_{w'})}{\sum_{\hat{w} \in V_i} \exp(\mathbf{h}^T \mathbf{v}_{\hat{w}})} \nabla_\theta \mathbf{h}^T \mathbf{v}_w'.
\end{aligned}$$

noise constrastive estimation (gutmann and hyvarinen 2012)

Distinguish real samples from noise using logistic regression

- Use uniform or *flattened* unigram as noise distribution Q
- Given k samples from Q and 1 from empirical distribution $\tilde{p}$ conditional is

$$\Pr(D = 0|\mathbf{h}, w) = \frac{kQ(w)}{\tilde{p}(w|\mathbf{h}) + kQ(w)}$$

  and probability of true data as

$$\Pr(D = 1|\mathbf{h}, w) = \frac{\tilde{p}(w|\mathbf{h})}{\tilde{p}(w|\mathbf{h}) + kQ(w)}.$$

noise constrastive estimation (gutmann and hyvarinen 2012)

Now maximize likelihood of this artificial corpus under model assuming it's *self-normalized*, i.e. $Z_{\mathbf{h}}$ is 1 for all $\mathbf{h}$

$$\mathsf{Pr}(D = 0|\mathbf{h}, w) = \frac{kQ(w)}{\exp(\mathbf{h}^T w) + kQ(w)}$$

and

$$\mathsf{Pr}(D = 1|\mathbf{h}, w) = \frac{\exp(\mathbf{h}^T w)}{\exp(\mathbf{h}^T w) + kQ(w)}.$$

Or rather solve logistic regression problem with same parameters (see Dyer 2014).

negative sampling (mikolov et al. 2013)

Can be seen as an approximation of noise contrastive estimation where

$$\Pr(D = 0|\mathbf{h}, w) = \frac{1}{\exp(\mathbf{h}^T w) + 1}$$

and probability of true data as

$$\Pr(D = 1|\mathbf{h}, w) = \frac{\exp(\mathbf{h}^T w)}{\exp(\mathbf{h}^T w) + 1}.$$

Again optimize parameters to do binary classification.

negative sampling (mikolov et al. 2013)

Can be seen as an approximation of noise contrastive estimation where

$$\Pr(D = 0|\mathbf{h}, w) = \frac{1}{\exp(\mathbf{h}^T w) + 1}$$

and probability of true data as

$$\Pr(D = 1|\mathbf{h}, w) = \frac{\exp(\mathbf{h}^T w)}{\exp(\mathbf{h}^T w) + 1}.$$

Again optimize parameters to do binary classification.

Sharing noise samples across all words in a mini-batch makes NCE and NS GPU friendly.
(See also BlackOut (Ji et al. 2015))

OPEN VOCABULARY NMT

vocabulary restrictions: computational and statistical

- Input and output embedding layer (typically) have most parameters.
- Vocabulary size determines complexity of softmax.
- Typically NMT systems limit vocabulary to < 50,000 (cf. 1,000,000 in phrase-based MT)

Difficult to translate names, places, numbers etc.

handling out-of-vocabulary (oov) words

Simple pre-processing/post-processing technique:

1. Map OOV source words to ’<unk>’.
    - Chelsea - Barcelone qui est favori ?
    - <unk> - <unk> est favori ?

handling out-of-vocabulary (oov) words

Simple pre-processing/post-processing technique:

1. Map OOV source words to ’<unk>’.
   - Chelsea - Barcelone qui est favori ?
   - <unk> - <unk> est favori ?
2. Include ’<unk>’ in target vocabulary.
   - <unk> - <unk> who is the favorite ?

handling out-of-vocabulary (oov) words

Simple pre-processing/post-processing technique:

1. Map OOV source words to '<unk>'.
   - Chelsea - Barcelone qui est favori ?
   - <unk> - <unk> est favori ?
2. Include '<unk>' in target vocabulary.
   - <unk> - <unk> who is the favorite ?
3. Find source of '<unk>' from attention weights
4. Copy or use word-level translation to post-process.
   - Chelsea - Barcelona who is the favourite ?

handling out-of-vocabulary (oov) words

Simple pre-processing/post-processing technique:

1. Map OOV source words to '<unk>'.
    - Chelsea - Barcelone qui est favori ?
    - <unk> - <unk> est favori ?
2. Include '<unk>' in target vocabulary.
    - <unk> - <unk> who is the favorite ?
3. Find source of '<unk>' from attention weights
4. Copy or use word-level translation to post-process.
    - Chelsea - Barcelona who is the favourite ?

Why might this not work well?

handling out-of-vocabulary (oov) words

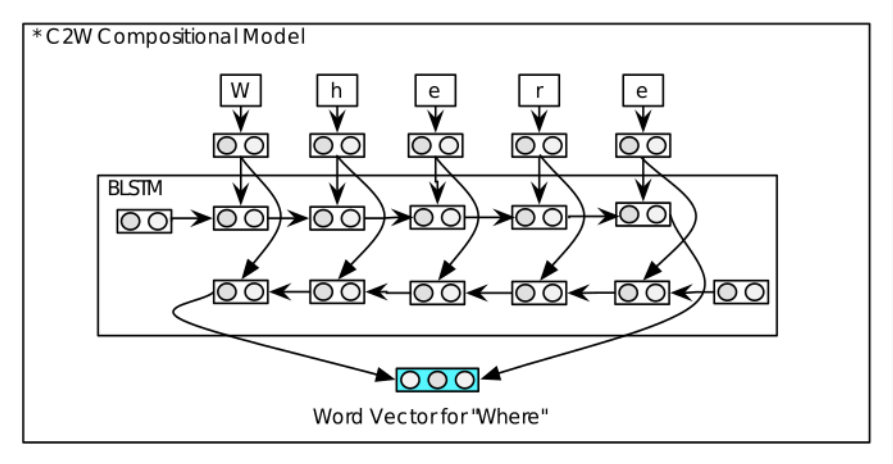
Simple pre-processing/post-processing technique:

1. Map OOV source words to ’<unk>’.
   - Chelsea - Barcelone qui est favori ?
   - <unk> - <unk> est favori ?
2. Include ’<unk>’ in target vocabulary.
   - <unk> - <unk> who is the favorite ?
3. Find source of ’<unk>’ from attention weights
4. Copy or use word-level translation to post-process.
   - Chelsea - Barcelona who is the favourite ?

Why might this not work well?

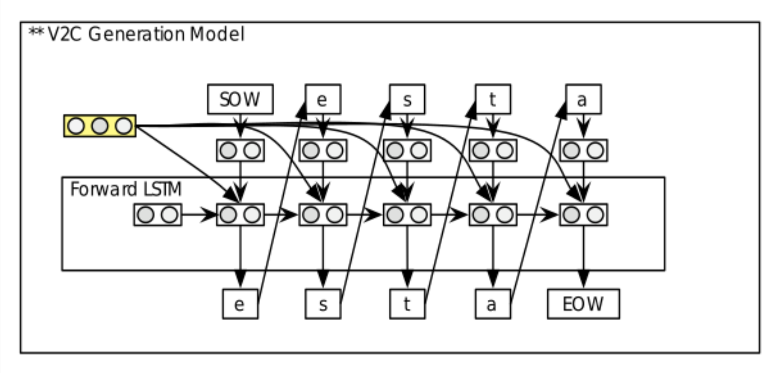
<unk> - <unk> <unk> <unk> <unk> ?

character-based nmt (ling et al. 2016)



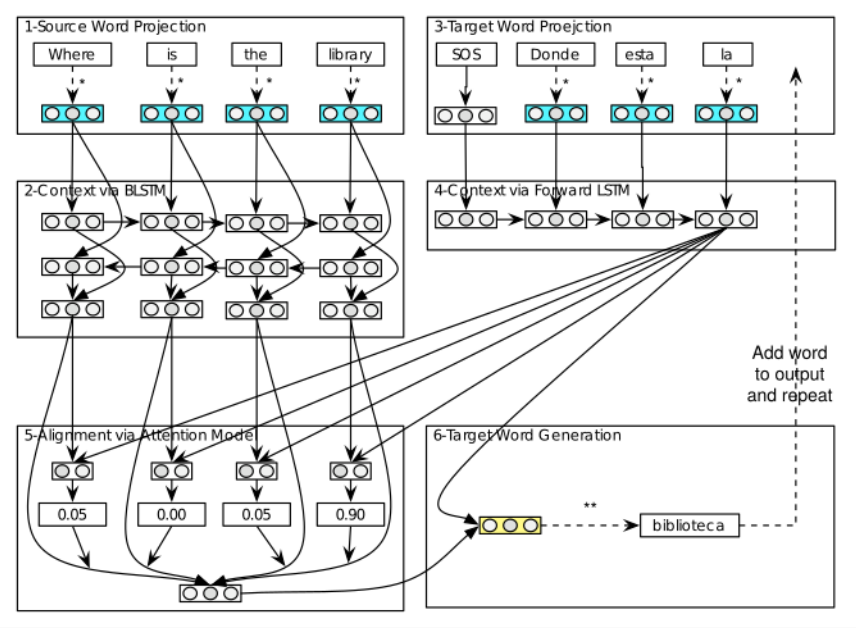
Replaces the input word embedding matrix

character-based nmt (ling et al. 2016)



Character generation model conditions on previous characters, aligned source words and target state.

character-based nmt (ling et al. 2016)



character-based nmt (ling et al. 2016)

- Predicts words from characters (still needs segmentation).

character-based nmt (ling et al. 2016)

- Predicts words from characters (still needs segmentation).
- Two beam searchs (one more for v2w model over words).

character-based nmt (ling et al. 2016)

- Predicts words from characters (still needs segmentation).
- Two beam searchs (one more for v2w model over words).
- v2c output layer is more efficient than softmax (?)

character-based nmt (ling et al. 2016)

- Predicts words from characters (still needs segmentation).
- Two beam searchs (one more for v2w model over words).
- v2c output layer is more efficient than softmax (?)
- But c2w slows down training significantly.

character-based nmt (ling et al. 2016)

- Predicts words from characters (still needs segmentation).
- Two beam searchs (one more for v2w model over words).
- v2c output layer is more efficient than softmax (?)
- But c2w slows down training significantly.
- Layer-wise training:
  1. Replace c2w layer by word embedding matrix.
  2. Learn this matrix, attention and v2c models first.
  3. Then train c2w to reproduce word embeddings.
  4. Finally train full model: c2w, attention, v2c.

character-based nmt (ling et al. 2016)

- Predicts words from characters (still needs segmentation).
- Two beam searchs (one more for v2w model over words).
- v2c output layer is more efficient than softmax (?)
- But c2w slows down training significantly.
- Layer-wise training:
  1. Replace c2w layer by word embedding matrix.
  2. Learn this matrix, attention and v2c models first.
  3. Then train c2w to reproduce word embeddings.
  4. Finally train full model: c2w, attention, v2c.

Use IBM model 4 alignments to constraint attention model.

sub-word models (sennrich et al. 2015)

- Find a trade off between word and character models
- Byte pair encoding algorithm (Gage, 1994):
  - Replace most frequent pair of bytes with a single code.

word	freq
'low </w>'	5
'low e r </w>'	2
'n e w est</w>'	6
'w i d est</w>'	3

freq	symbol pair		new symbol
9	('e', 's')	→	'es'
9	('es', 't')	→	'est'
9	('est', '</w>')	→	'est</w>'
7	('l', 'o')	→	'lo'
7	('lo', 'w')	→	'low'
...			

(From Sennrich et al. 2016)

byte pair encoding

Popular choice in current NMT systems

- Can be used together with UNK-replacement.
- Doesn't waste time on common sequences (cf. character models).
- Common to use between codebook of 10k-80k sub-words.

Used in the baseline systems for Assignment 4.

OTHER TOPICS

monolingual data

- Would be nice to use all that monolingual data

monolingual data

- Would be nice to use all that monolingual data
- Approaches include:
  - Use language model to rescore candidates (Gulccehre et al. 2015).

monolingual data

- Would be nice to use all that monolingual data
- Approaches include:
  - Use language model to rescore candidates (Gulccehre et al. 2015).
  - Create pseudo-parallel corpus by translating target corpus to source language (Sennrich et al. 2015).

monolingual data

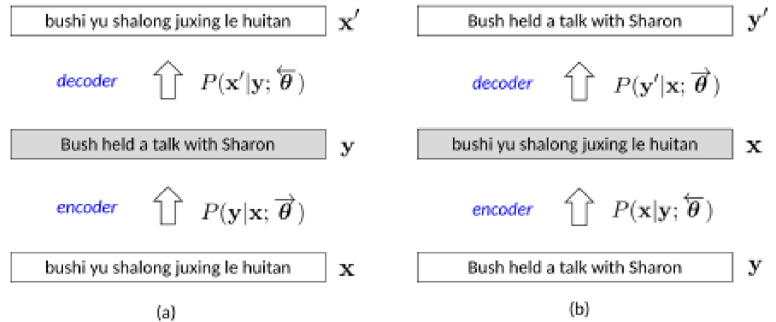
- Would be nice to use all that monolingual data
- Approaches include:
  - Use language model to rescore candidates (Gulccehre et al. 2015).
  - Create pseudo-parallel corpus by translating target corpus to source language (Sennrich et al. 2015).
  - Semi-supervised learning: train source-to-target and target-to-source models and use to build autoencoders for monolingual source and target corpora (Cheng et al. 2016).

monolingual data

- Would be nice to use all that monolingual data
- Approaches include:
  - Use language model to rescore candidates (Gulccehre et al. 2015).
  - Create pseudo-parallel corpus by translating target corpus to source language (Sennrich et al. 2015).
  - Semi-supervised learning: train source-to-target and target-to-source models and use to build autoencoders for monolingual source and target corpora (Cheng et al. 2016).

But LM might be less important than in PBMT. Why?

semi-supervised learning for nmt (cheng et al. 2016)



Can use source (a) and and target (b) monolingual corpora.

sequence training

- Mismatch between training and test time: target context might be incorrect at test time
- Tune pre-trained model by sampling target contexts rather than using reference contexts.

(See Shen et al. 2016)

ensembles

- Local optima are common when training RNNs.
- Combine distributions of N models at each time step (requires same output vocabulary).
- Arithmetic mean of log probs (OR) or geometric mean (AND)
  - Different architectures
  - Different hyperparameters
  - Different checkpoints
  - Different source languages (!)

REFERENCES

references (1)

- Bahdanau et al. 2015, Neural machine translation by jointly learning to align and translate.
- Bentivogli 2016: Neural versus Phrase-Based Machine Translation Quality: a Case Study.
- Cheng et al. 2016: Semi-Supervised Learning for Neural Machine Translation.
- Ji et al. 2015: BlackOut: Speeding up recurrent neural network language models with very large vocabularies.

references (2)

- Ling et al. 2016: Character-based Neural Machine Translation, ICLR 2016.
- Mi et al. 2016: Coverage Embedding Models for Neural Machine Translation.
- Neubig 2017: Draft book chapters on NMT (Neural MT 1: Encoder-Decoder Models, Neural MT 2: Attentional Models).
- Sennrich et al. 2015: Neural Machine Translation of Rare Words with Subword Units.