Machine Translation and Neural Networks

Gaurav Kumar
Center for Language and Speech Processing
Johns Hopkins University
gkumar@cs.jhu.edu

03/03/2015

Machine Translation : The Generative Story

- Given a source sentence $\mathbf{f}$, we want to find the most likely translation $\mathbf{e}^*$

$$
\begin{aligned}
e^* &= \arg\max_{\mathbf{e}} p(\mathbf{e}|\mathbf{f}) \\
&= \arg\max_{\mathbf{e}} p(\mathbf{f}|\mathbf{e})\ p(\mathbf{e}) && \text{(Bayes Rule)} \\
&= \arg\max_{\mathbf{e}} \sum_{\mathbf{a}} p(\mathbf{f}, \mathbf{a}|\mathbf{e})\ p(\mathbf{e}) && \text{(Marginalize over alignments)}
\end{aligned}
$$

- The alignments $\mathbf{a}$ are latent. $p(\mathbf{f}, \mathbf{a}|\mathbf{e})$ is typically decomposed as:
  - Lexical/Phrase **Translation Model**
  - An **Alignment/Distortion Model**

- $p(\mathbf{e})$ is the **Language Model**

Machine Translation : Additional Features

- Decoding may find features besides the ones derived from the generative model useful
  - reordering (distortion) model
  - phrase/word translation model
  - language models
  - word count
  - phrase count
- In phrase based models, how do you explicitly measure the quality of a phrase pair ?
- Weights are typically tuned on a *development* set using discriminative training.

Neural Networks and Machine Translation

- The use of neural networks has been proposed for almost all components of machine translation.

- We will look at three propositions today. One for each of the following:
  - **Language Models**

$$p(e_i|e_1 \cdots e_{i-1})$$

  - **Additional features for machine translation**

$$p(\mathbf{e}|\mathbf{f}) = \frac{\sum_i \lambda_i \; k_i}{Z} \qquad \text{(a feature } k_i \text{ has a weight } \lambda_i\text{)}$$

  - **Translation and Alignment models**

$$p(\mathbf{f}, \mathbf{a}|\mathbf{e})$$

Neural Language Models

Neural Language Models

- **Neural Network Joint Model (NNJM)** (*Devlin et al., ACL 2014*)
  - Extends the neural network language models (NNLM) (*Bengio et al., 2003; Schwenk, 2010*)
  - Incorporates source side context in language models
  - Requires parallel text with alignments to train
  - Speedup tricks makes querying as fast as backoff LMs

- **Main Idea** : Incorporate source side context

$$p(\mathbf{e}, \mathbf{a}|\mathbf{f}) \approx \prod_{i=1}^{|\mathbf{e}|} p(e_i|e_{i-1} \cdots e_{i-n+1}, \mathcal{F}_i)$$

Where $\mathcal{F}_i$ is the source context vector

Neural Network Joint Model (NNJM)

- **Main Idea** : Incorporate source side context

$$p(\mathbf{e}, \mathbf{a}|\mathbf{f}) = p(\mathbf{e}|\mathbf{f}) \approx \prod_{i=1}^{|\mathbf{e}|} p(e_i|e_{i-1} \cdots e_{i-n+1}, \mathcal{F}_i)$$

  - Where $\mathcal{F}_i$ is the source context vector
  - $\mathbf{a}$ is a deterministic function of $\mathbf{e}$ and $\mathbf{f}$
  - Use a source context window around f_{a_i}.

S:	我	[3] 就	[4] 取	[5] 钱	[6] 给	[7] 了	她们
	i	*will*	*get*	*money*	*to*	perf.	*them*

T: [2] i [1] will [0] get the money to them

P(the | get, will, i, 就, 取, 钱, 给, 了)

  - This is effectively an $(n + m)$-gram language model.

Neural Network Joint Model (NNJM) : Training

- A feed-forward neural network is used (two hidden layers)
- The input is the concatenated word embeddings for the $((n-1)+m)$ context vector
- OOVs are mapped to their POS tags (special OOV tag when no POS tag is available)
- Training is done using back-propagation with the maximization of the log-likelihood of the training data as the objective

$$L = \sum_i \log(p(x_i))$$

where x_i is one training sample.

Speedup Trick : Normalization

- A softmax over the entire target vocabulary is expensive

$$p(x) = \frac{e^{U_r(x)}}{\sum_{\mathbf{r'=1}}^{\mathbf{|V_t|}} \mathbf{e^{U'_r(x)}}}$$

where $U_r(x)$ is the activated value of the output layer corresponding to the observed target word and V_t is the length of the target vocabulary

- **Main Idea** : Force $Z(x)$ to be close to 1 by augmenting the objective function

$$L = \sum_i \left[\log(p(x_i)) - \alpha \log^2(Z(x_i))\right]$$

  – Maximizing this objective will encourage $\log^2(Z(x_i))$ to have values close to 0.
  – α is a parameter that can be tuned for a trade-off between accuracy and mean normalization error.

Speedup Trick : Pre-computing first hidden layer

- Use the fact that this is an $(n-1)+m$-gram model.
- A target word can be in one of $(n-1)$ positions.
- A source word can be in one of m positions.
- **Main Idea :** The dot product of each word in each position contributes a constant value to the hidden layer.
- Pre-compute the contributions and store them. Total number of pre-computations :

$$[(n-1) \times |V_t| + m \times |V_s|]$$

- Computing the first hidden later requires only a lookup for a word in a position now.

Additional features for Machine Translation

Phrasal Similarity

Features based on phrase similarity

Why can't you trust (all) phrase pairs?

- **Rare phrases**: Rare phrase pair occurrences provide a sub-optimal estimate for phrase translation probabilities.
  $p(\text{sorona} \mid \text{tristifical}) = 1$
  $p(\text{tristifical} \mid \text{sorona}) = 1$

- **Independence assumptions** : The choice to use one phrase pair over an another is largely independent of previous decisions.

- **Segmentation** : Phrase segmentation is generally not linguistically motivated and a large percentage of the phrase pairs are not good translations.
  (!, veinte dlares, era, you! twenty dollars, it was)
  (Exactamente como , how they want to)

- More information about phrases is (almost) always good.

Features based on phrase similarity

- Bilingual Constrained Recursive Autoencoders (BRAE) (*Zhang et al., ACL, 2014*)
  - Extends the use of unsupervised recursive encoders for phrase embedding (*Socher et al., Li et al., 2013*)
  - **Main Idea :** Find an embedding for each source phrase such that its embedding is close to the one for the corresponding target phrase (via transformation).

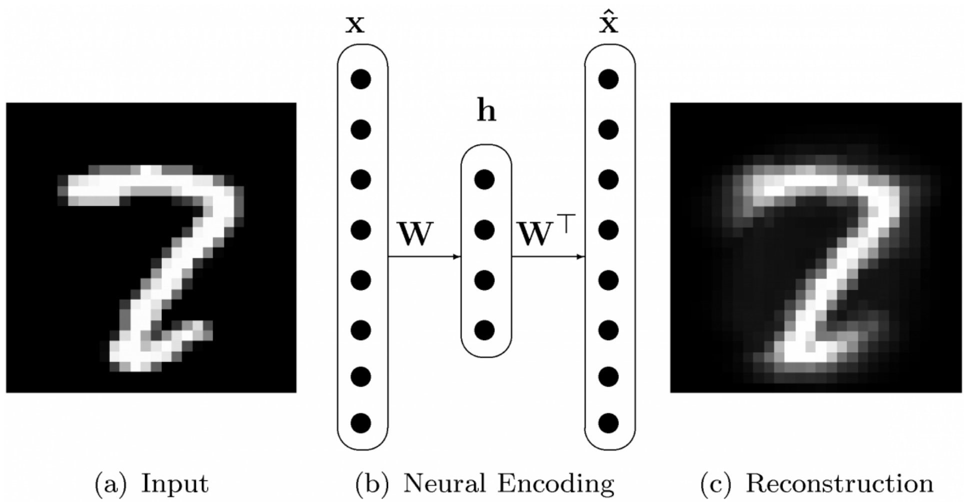
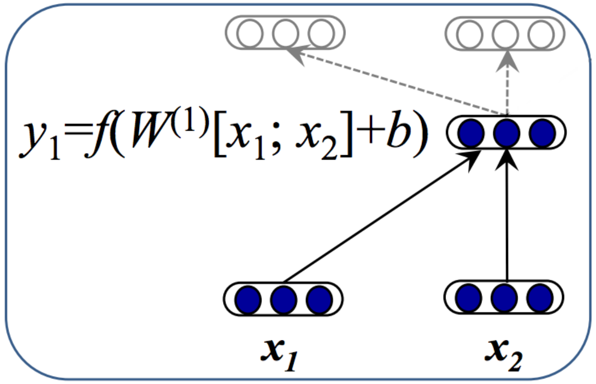


Figure 1: An autoencoder (Image from Lemme et al., 2010)

Phrase Embedding with Autoencoders



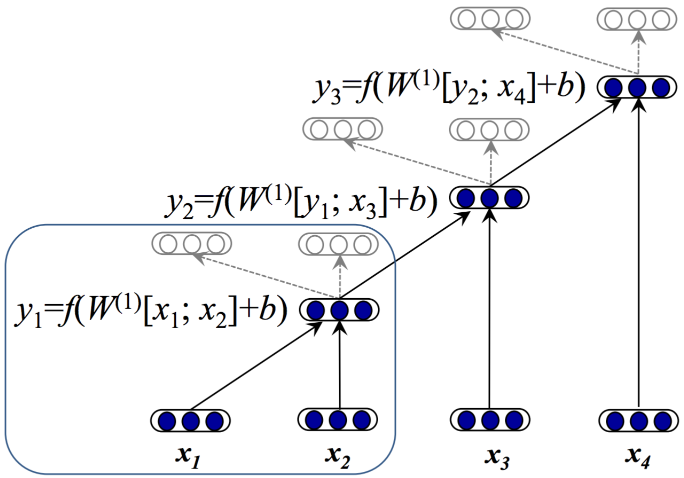
- Given two child vectors $c_1 = x_1$ and $c_2 = x_2$, the parent vector can be computed as

$$p = f(W^{(1)}[c_1; c_2] + b^{(1)})$$

- and the children can be reconstructed as

$$[c_1'; c_2'] = f(W^{(2)}p + b^{(2)})$$

Phrase Embedding with RAE



Phrase embedding with **Recursive** autoencoders

- Multi-word phrase
- Combine two leaves using the **same** autoencoder
- Continue for a binary tree until only one node (the root) remains.
- The root represents the embedding for the phrase

Phrase Embedding with RAE

- The error of reconstruction for one example

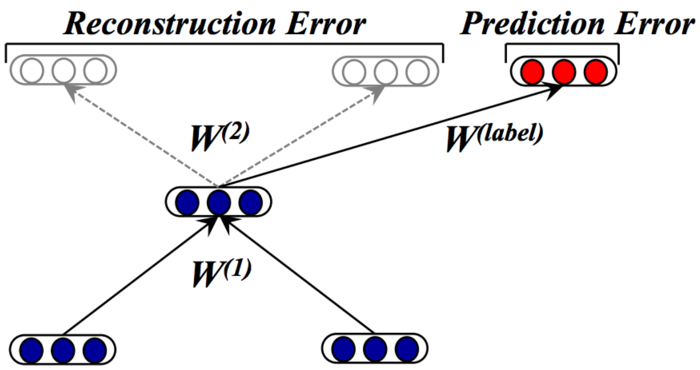
$$E_{rec}([c_1; c_2]) = \frac{1}{2}||[c_1; c_2] - [c'_1; c'_2]||^2$$

- The goal is to minimize this reconstruction error at each node for the optimal binary tree (for one phrase x)

$$RAE_\theta(x) = \arg\min_{y \in A(x)} \sum_{s \in y} E_{rec}([c_1; c_2]_s)$$

where $A(x)$ is the set of all binary trees for this phrase.

Autoencoders for Multi-Objective Learning



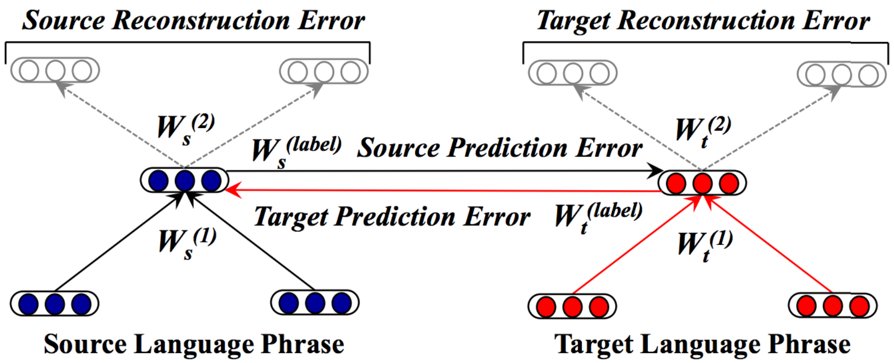
- A RAE can be used to predict a target label
  - Polarity in sentiment analysis (*Socher et al., 2011*)
  - Syntactic category in parsing (*Socher et al., 2013*)
  - Phrase reordering pattern for SMT (*Li et al., 2013*)

- Given a phrase and a label (x, t) the error becomes

$$E(x, t; \theta) = \alpha E_{rec}(x, t; \theta) + (1 - \alpha) E_{pred}(x, t; \theta)$$

where α is the interpolation hyper-parameter.

Bilingual Constrained Recursive Autoencoders



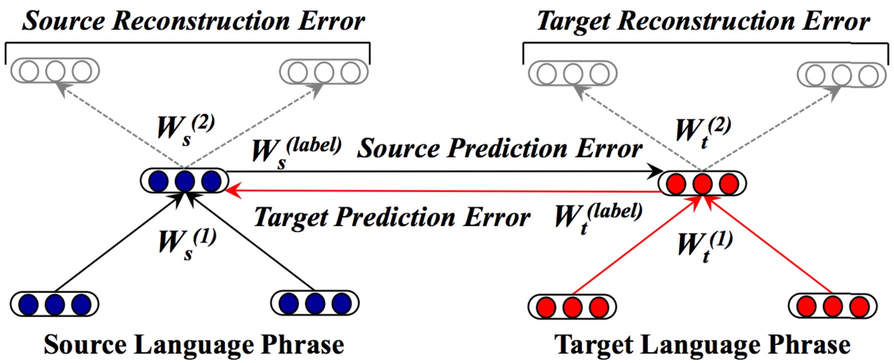
- For a phrase pair (s, t)
  - The reconstruction error is

$$E_{rec}(s, t; \theta) = E_{rec}(s; \theta) + E_{rec}(t; \theta)$$

  - The semantic error is

$$E_{sem}(s, t; \theta) = E_{sem}(s|t; \theta) + E_{sem}(t|s; \theta)$$

Bilingual Constrained Recursive Autoencoders



- The semantic error $E_{sem}(s|t;\theta)$ can be computed as

$$E_{sem}(s|t;\theta) = \frac{1}{2}||p_t - f(W_s^l p_s + b_s^l)||^2$$

- For each phrase pait (s, t) the joint error is

$$E(s,t;\theta) = \alpha E_{rec}(s,t;\theta) + (1-\alpha)E_{sem}(s|t;\theta)$$

BRAE : Phrasal similarity

- Given any phrase pair (s, t) this trained model can compute
  - The similarity between the transformed source and the target $Sim(p_s*, p_t)$
  - The similarity between the transformed target and the source $Sim(p_t*, p_s)$

- These can be used as :
  - Features to prune the phrase table
  - Features for discriminative training in phrase based SMT

Joint Alignment and Translation

Learning to align and translate

Joint learning of alignment and translation (*Bahdanau et al., 2015*)

- One model for translation and alignment
- Extends the standard RNN encoder-decoder framework for neural network based machine translation
- Allows the use of an alignment based soft search over the input
- In the presence of a deterministic alignment, this model simplifies into a translation model

RNN encoder-decoder

- **Encoder** : Given any sequence of vectors $(f_1, \cdots, f_J)$

$$s_j = r(f_j, s_{j-1}) \qquad \text{(Hidden state)}$$

$$c = q(\{s_1, \cdots, s_J\}) \qquad \text{(The context vector)}$$

where $s_j \in \mathbb{R}^n$ is the hidden state at time j, c is the context vector generated from the hidden states and r and q are some non-linear functions.

- **Decoder** : Predict e_i given $e_1, \cdots, e_{i-1}$ and the context c.

$$p(\mathbf{e}) = \prod_{i=1}^{I} p(e_i | \{e_1, \cdots, e_{i-1}\}, c) \qquad \text{(Joint probability)}$$

$$p(e_t | \{e_1, \cdots, e_{i-1}\}, c) = g(e_{i-1}, t_i, c) \qquad \text{(Conditional probability)}$$

where t_i is the hidden state of the RNN and g is some non-linear function that outputs a probability.

Joint alignment and translation : Decoder

- The conditional probability is now defined as

$$p(e_i|\{e_1, \cdots, e_{i-1}\}, c) = g(e_{i-1}, t_i, c_i)$$

where $t_i = g(t_{i-1}, e_{i-1}, c_i)$ is the hidden state.

- The context vector depends on representations that the encoder maps the input sentence to. $(f_j \rightarrow h_j)$

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

where the weight α_{ij} is calculated as

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k-1}^{T_x} \exp(e_{ik})}$$

and $e_{ij} = a(t_{i-1}, h_j)$ is the alignment model.

Joint alignment and translation : Decoder

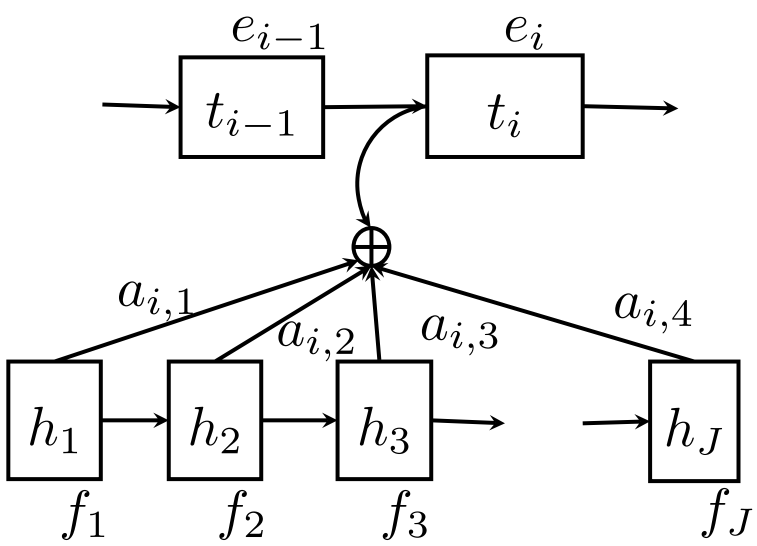


Figure 2: The hidden states depend on the input representations weighted by how well they align with the target word

Joint alignment and translation : Encoder

- We want the representation for each word to contain information about the forward and the backward context.

- Use Bi-directional RNNs where
  - The forward RNN $\overrightarrow{N}$ reads $\{f_1, \cdots, f_J\}$ and generates $\{\overrightarrow{h_1}, \cdots, \overrightarrow{h_J}\}$
  - The backward RNN $\overleftarrow{N}$ reads $\{f_J, \cdots, f_1\}$ and generates $\{\overleftarrow{h_1}, \cdots, \overleftarrow{h_J}\}$

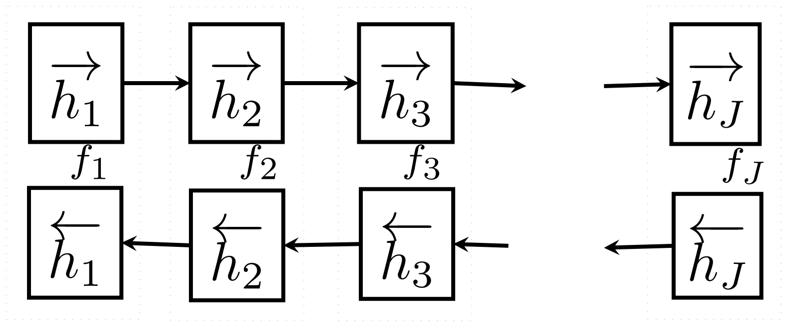


Figure 3: Concatenate forward and backward hidden states to obtain the representation for each word.

Joint alignment and translation : Decoder

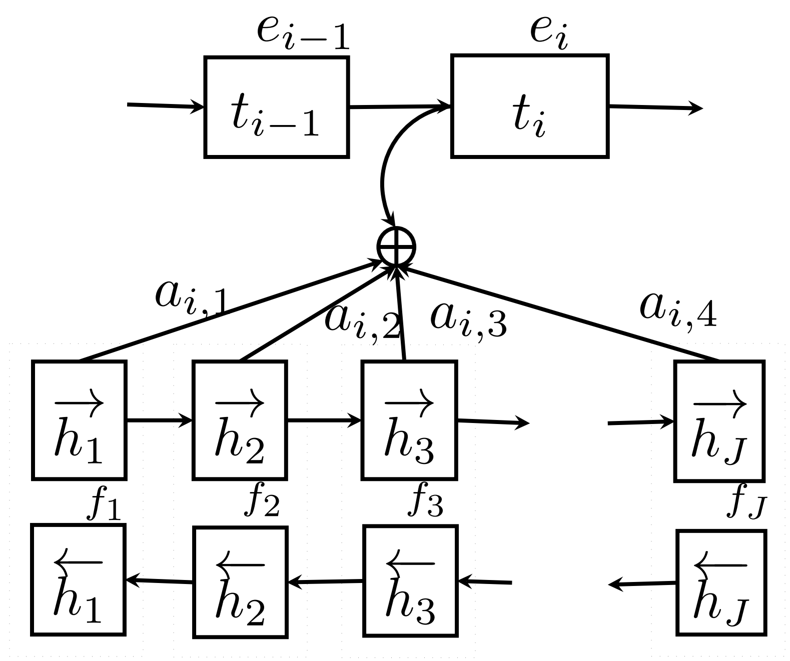


Figure 4: Putting it all together : The annotations created by concatenating the hidden states are used by the decoder

Conclusion

How well do these models perform ?

- **NNJM** uses source side context along with the target side.
  - $+3.0$ BLEU gain over a state of-the-art S2T system with NNLM.
  - $+6.0$ BLEU gain over a simple hierarchical system with regular n-gram LMs.

- **BRAE** adds additional features which describe phrasal similarity to an existing translation model.
  - Reduced loss in translation quality while pruning compared to Significance pruning.

- The **joint-alignment-translation RNN** describes one self-sufficient system for alignment and translation.
  - Results comparable with current phrase based systems.

Acknowledgments

- Philipp Koehn for the slide template.

- Yuan Cao, Sanjeev Khudanpur and Philipp Koehn for feedback on content and structure.

- The MT@JHU reading group for the ideas.