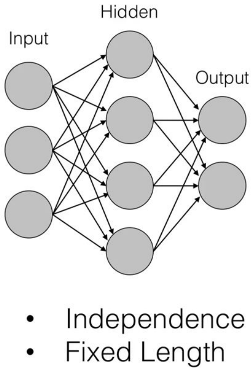
Sequence Modeling: Recurrent and Recursive Networks

Markus Dumke

27th January 2016

Contents

Why RNN's?



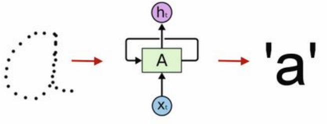
https://www.nervanasys.com/recurrent-neural-networks/

He went to Germany in 2010.

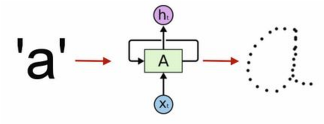
In 2010 he went to Germany.

- sequential data:
  texts, speech, time series
- variable length
- long-term dependencies
- memory

Applications



Handwriting recognition



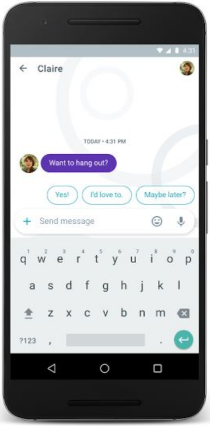
Handwriting generation

https://greydanus.github.io/2016/08/21/handwriting/

Applications

"man in black shirt is playing guitar."

Image Captioning



Smart reply

cs.stanford.edu/people/karpathy/deepimagesent/

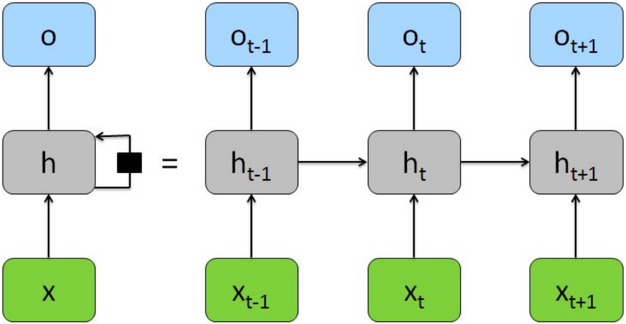
https://research.googleblog.com/2016/05/chat-smarter-with-allo.html

Applications

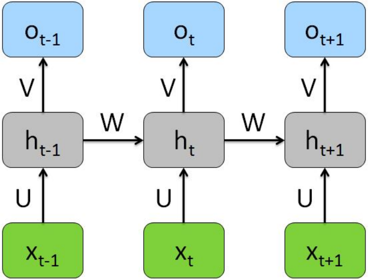
- Machine translation
- Sentiment analysis
- Text summaries
- Speech recognition and generation
- Time series
- Deep Reinforcement Learning

Contents

Recurrent Neural Network



Recurrent Neural Network

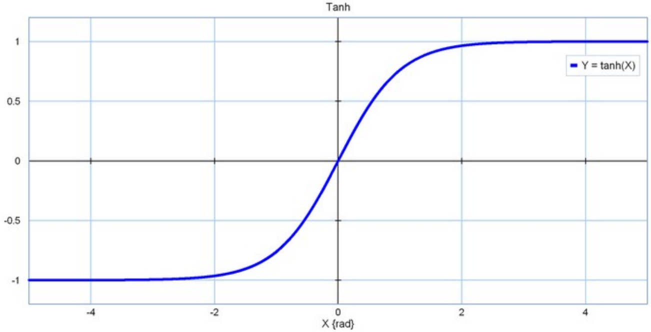


for t = 1 to T:

$$
\begin{aligned}
h_t &= f(b + W\, h_{t-1} + U\, x_t) \\
o_t &= c + V\, h_t \\
\hat{y}_t &= softmax(o_t) \\
&= \frac{exp(o_t^{k'})}{\sum_k exp(o_t^k)} \quad \forall k'
\end{aligned}
$$

Which activation function?

$$f(x) = tanh(x) = \frac{sinh(x)}{cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



http://www.20sim.com/webhelp/language_reference_functions_tanh.php

Language Modeling

- Input: word/character encoded as one-hot vector
- Output: Probability distribution over words given previous words

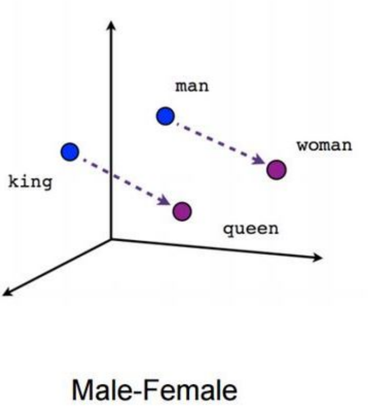
$$\mathrm{P}(\mathrm{y}_1, ..., y_T) = \prod_{i=1}^{T} P(y_i | y_1, ..., y_{i-1})$$

- scoring candidates

Word embeddings (Word2vec)

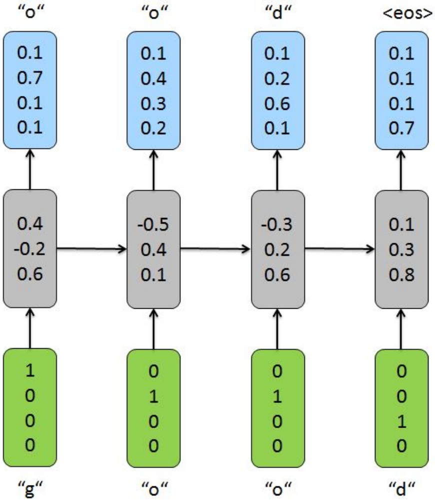
- Data sparsity

$$man \rightarrow \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \rightarrow \begin{bmatrix} 0.35 \\ 0.83 \\ \vdots \\ 0.34 \\ 0.66 \end{bmatrix}$$



https://www.tensorflow.org/tutorials/word2vec/

Recurrent Neural Network



Sampling from an RNN

- Sample from conditional distribution at each time step
- How to generate sequence length?
  - special end symbol
  - Bernoulli random variable
  - integer value τ

Contents

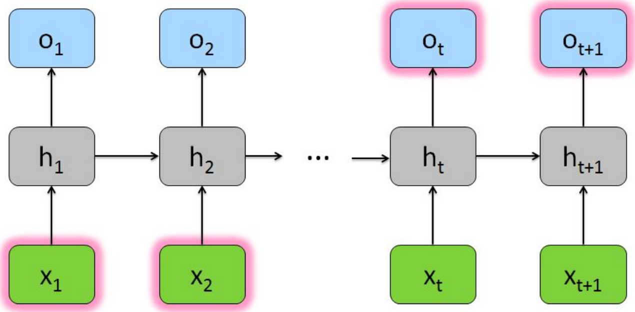
Optimization

- Forward Propagation:
  - compute hidden states, outputs and loss
  - Loss function, e.g. Bernoulli loss, MSE

$$L = \sum_t L_t$$

- Backward Propagation through time (BPTT):
  - compute gradients
- Stochastic Gradient Descent
  - Minibatch

Vanishing (and Exploding) Gradient Problem

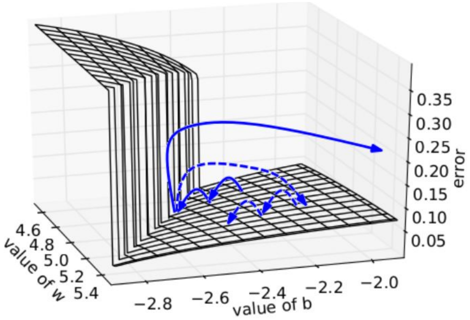


How to deal with exploding gradients?

Gradient Clipping

$$\textit{if } ||\nabla W|| > \textit{threshold} :$$

$$\nabla W \leftarrow \frac{\textit{threshold}}{||\nabla W||} \nabla W$$



http://www.jmlr.org/proceedings/papers/v28/pascanu13.pdf

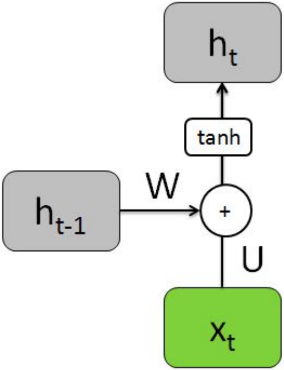
How to deal with vanishing gradients?

- Regularization $\nabla_{h_t} L \approx (\nabla_{h_t} L) \dfrac{\partial h_t}{\partial h_{t-1}}$
- skip-connections over time
- Leaky units $\mu = \alpha\, \mu_{t-1} + (1 - \alpha)\, \nu_t$
- remove short-term connections
- **LSTM**, GRU and other gated RNNs

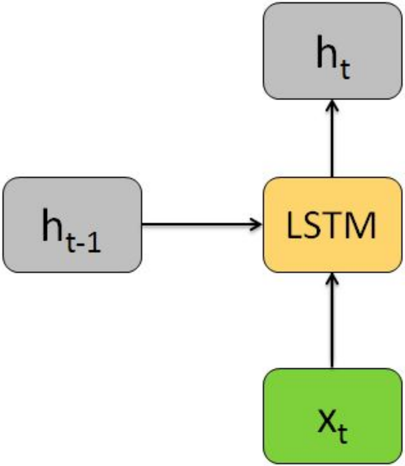
Contents

Vanilla RNN

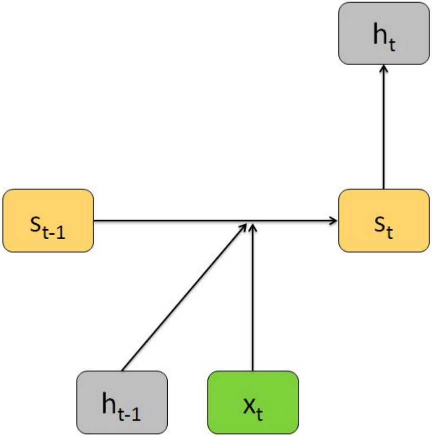
$$h_t = \tanh(b + W\, h_{t-1} + U\, x_t)$$



LSTM



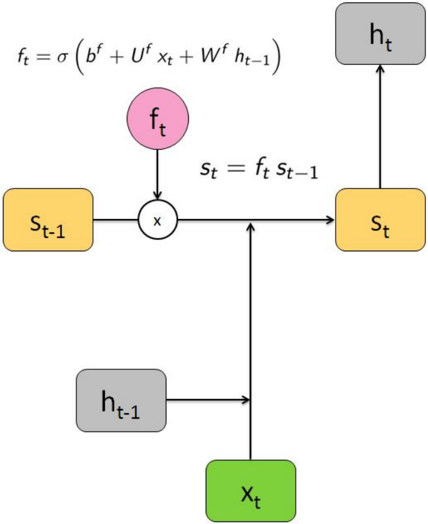
LSTM



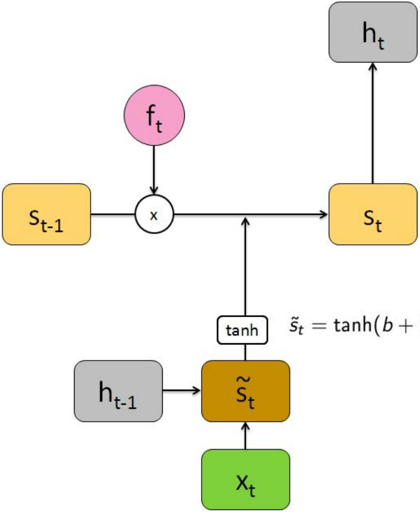
LSTM

$$f_t = \sigma\left(b^f + U^f x_t + W^f h_{t-1}\right)$$

$$s_t = f_t\, s_{t-1}$$

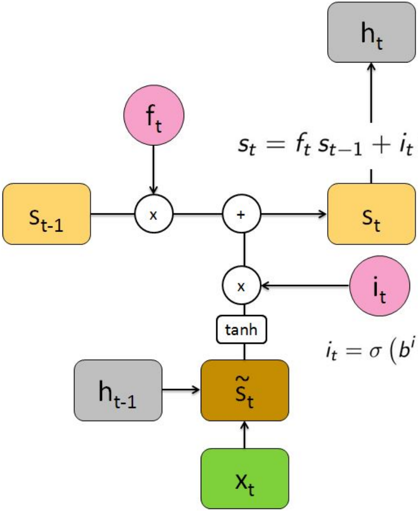


LSTM



$$\tilde{s}_t = \tanh(b + W\, h_{t-1} + U\, x_t)$$

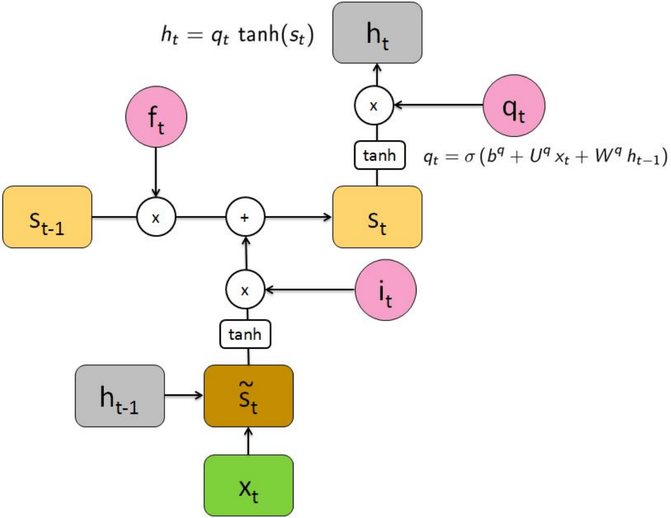
LSTM



$$s_t = f_t\, s_{t-1} + i_t\, \tilde{s}_t$$

$$i_t = \sigma\left(b^i + U^i\, x_t + W^i\, h_{t-1}\right)$$

LSTM



$h_t = q_t \tanh(s_t)$
h_t
x
q_t
f_t
tanh
$q_t = \sigma\left(b^q + U^q x_t + W^q h_{t-1}\right)$
s_{t-1}
x
+
s_t
x
i_t
tanh
h_{t-1}
$\tilde{s}_t$
x_t

LSTM in R (mxnet)

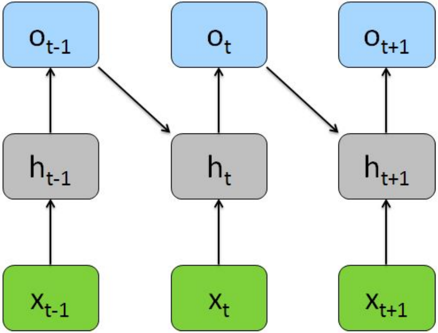
```
model <- mx.lstm(X.train, X.val,
                 ctx = mx.cpu(),
                 num.round = 100,
                 update.period = 1,
                 num.lstm.layer = 1,
                 seq.len = 32,
                 num.hidden = 16,
                 num.embed = 16,
                 num.label = 100,
                 batch.size = 32,
                 input.size = 100,
                 initializer = mx.init.uniform(0.1),
                 learning.rate = 0.1,
                 wd = 0.00001,
                 clip_gradient = 1)
```

Generating Text

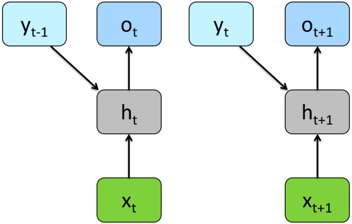
```
infer.model <- mx.lstm.inference(num.lstm.layer=num.lstm.layer,
                                 input.size=vocab,
                                 num.hidden=num.hidden,
                                 num.embed=num.embed,
                                 num.label=vocab,
                                 arg.params=model\$arg.params,
                                 ctx=mx.cpu())

mx.lstm.forward(infer.model, input, FALSE)
```

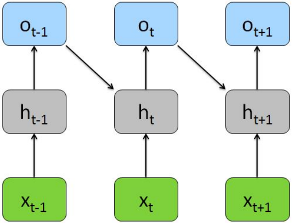
RNN with output recurrence



Teacher Forcing

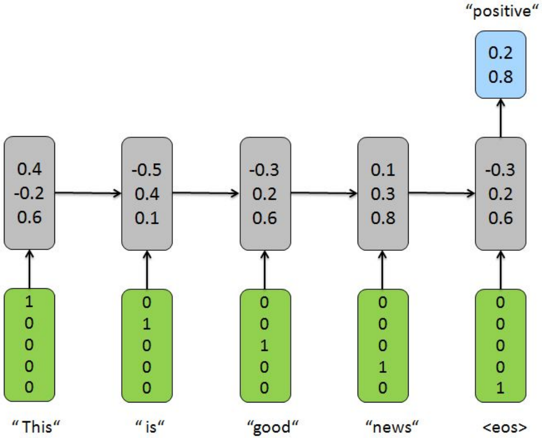


At train time

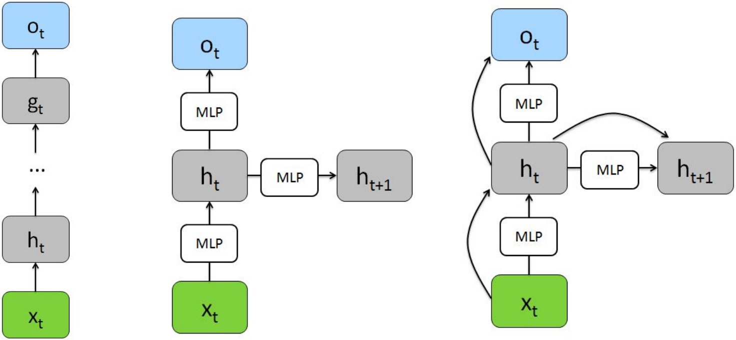


At test time

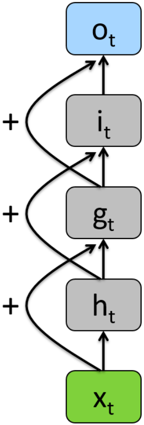
One-output RNN



Deep RNNs

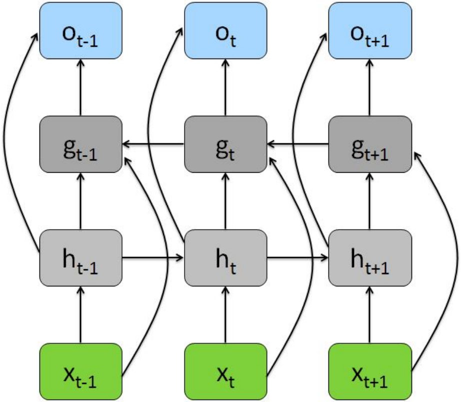


Residual Networks (Res-Nets)

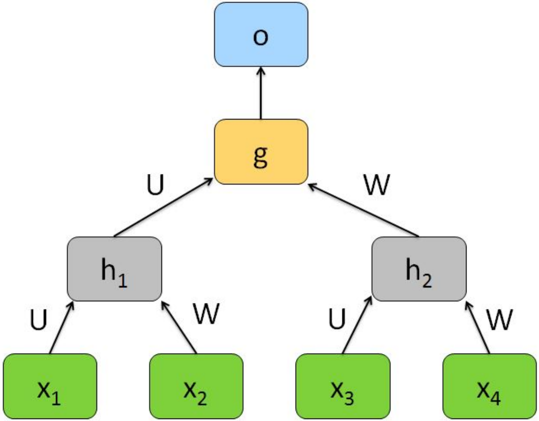


- training of very deep models possible
- like an ensemble of shallow architectures

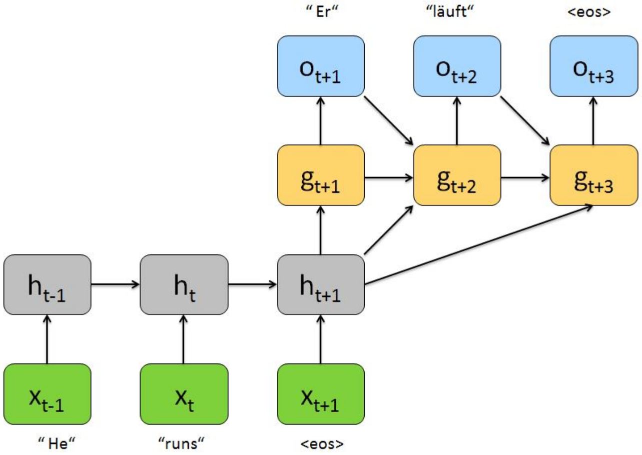
Bidirectional RNN



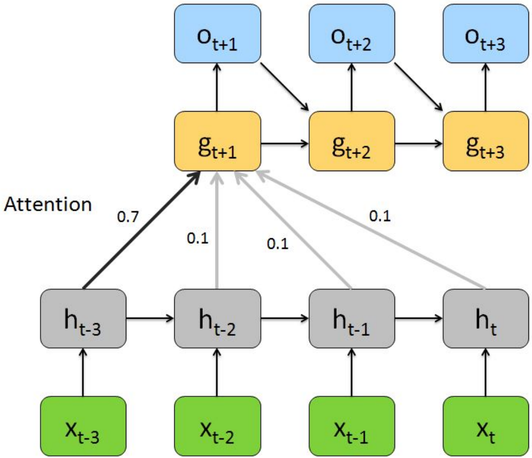
Recursive Neural Network



Encoder-Decoder Network

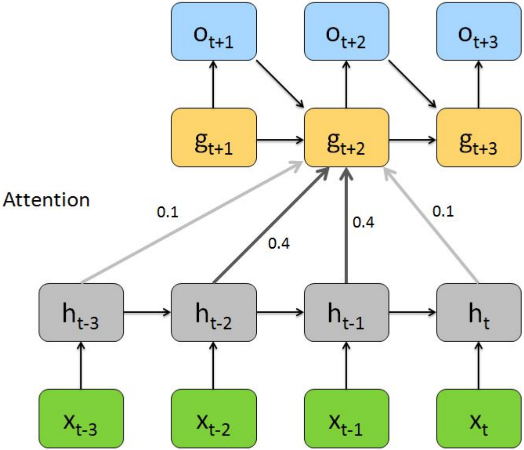


Attention



o_{t+1}
o_{t+2}
o_{t+3}
g_{t+1}
g_{t+2}
g_{t+3}
Attention
0.7
0.1
0.1
0.1
h_{t-3}
h_{t-2}
h_{t-1}
h_t
x_{t-3}
x_{t-2}
x_{t-1}
x_t

Attention



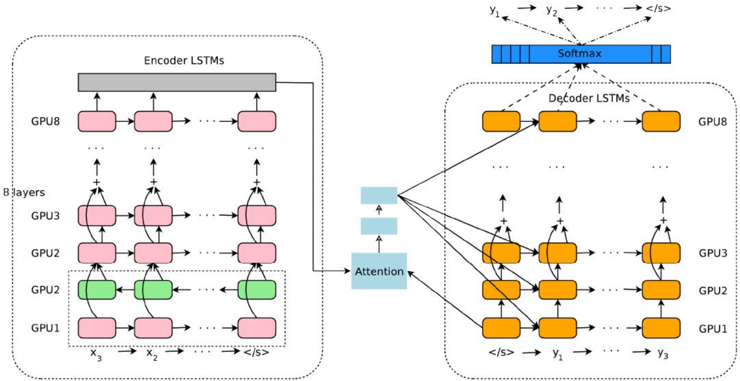
o_{t+1}
o_{t+2}
o_{t+3}
g_{t+1}
g_{t+2}
g_{t+3}
Attention
0.1
0.4
0.4
0.1
h_{t-3}
h_{t-2}
h_{t-1}
h_t
x_{t-3}
x_{t-2}
x_{t-1}
x_t

Contents

Machine Translation

- last decades: phrase-based systems
- neural networks as part of phrase-based systems
- Encoder-decoder RNNs:
  - Sutskever et al. (2014), Bahdanau et al. (2015)
- Google's Neural Machine Translation (September/November 2016)

Google's Neural Machine Translation System



Wu et al. (2016): Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation

Details

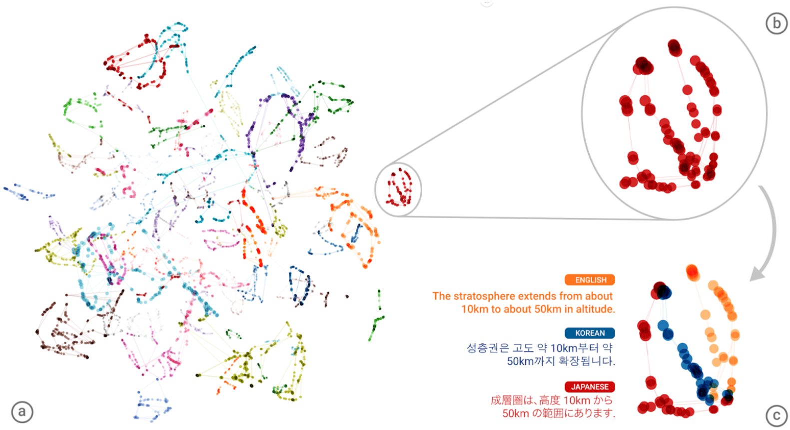
Main challenges:

- speed
- handling of rare words
- not translating all words (coverage)

Solutions:

- GPU training
- sub-word units (wordpieces)
- coverage penalty
- length-normalization

Language Embeddings



Johnson et al. (2016): Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation