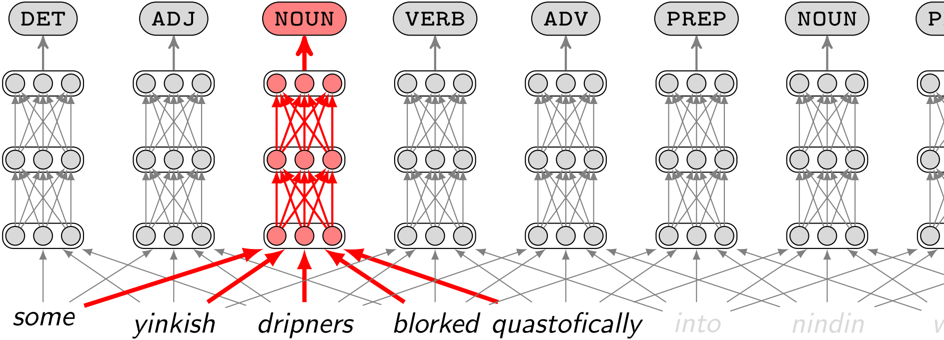
L90: Overview of Natural Language Processing

Lecture 6: Neural Networks

Weiwei Sun

Department of Computer Science and Technology
University of Cambridge

Michaelmas 2020/21



good features can be automatically induced

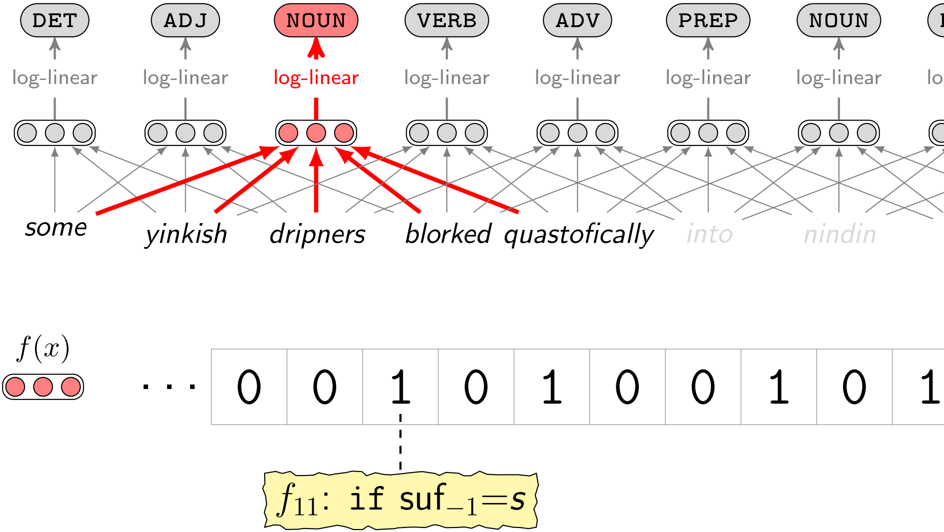
Lecture 6: Neural Networks

1. Feature engineering → Representation Learning
2. Feedforward Neural Networks
3. Advancing Dependency Parsing
4. Some General Comments on Neural NLP

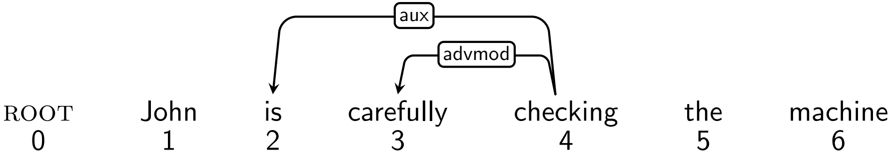
some slides are from Ann Copestake

Representation Learning

Recap: POS tagging and hand-crafted features



Hand-crafted features for dependency parsing



Stack

[ROOT, John]

Buffer/Queue

[checking, the, machine]

how is this configuration represented?

Feature template

from single words
S_0wp; S_0w; S_0p; N_0wp; N_0w; N_0p; N_1wp; N_1w; N_1p; N_2wp; N_2w; N_2p;
from word pairs
S_0wpN_0wp; S_0wpN_0w; S_0wN_0wp; S_0wpN_0p; S_0pN_0wp; S_0wN_0w; S_0pN_0p; N_0pN_1p;
from three words
$N_0pN_1pN_2p$; $S_0pN_0pN_1p$; $S_{0h}pS_0pN_0p$; $S_0pS_{0l}pN_0p$; $S_0pS_{0r}pN_0p$; $S_0pN_0pN_{0l}p$;

- S_i the i-th element in the stack
- N_i the i-th element in the buffer
- w word form
- p POS label
- l/r left/right most dependent

from Zhang and Nivre (2011)

Feature template (cont)

distance
S_0wd; S_0pd; N_0wd; N_0pd; S_0wN_0wd; S_0pN_0pd;
valency
S_0wv_r; S_0pv_r; S_0wv_l; S_0pv_l; N_0wv_l; N_0pv_l;
unigrams
$S_{0h}w$; $S_{0h}p$; S_0l; $S_{0l}w$; $S_{0l}p$; $S_{0l}l$; $S_{0r}w$; $S_{0r}p$; $S_{0r}l$; $N_{0l}w$; $N_{0l}p$; $N_{0l}l$;
third-order
$S_{0h2}w$; $S_{0h2}p$; $S_{0h}l$; $S_{0l2}w$; $S_{0l2}p$; $S_{0l2}l$; $S_{0r2}w$; $S_{0r2}p$; $S_{0r2}l$; $N_{0l2}w$; $N_{0l2}p$; $N_{0l2}l$; $S_0pS_{0l}pS_{0l2}p$; $S_0pS_{0r}pS_{0r2}p$; $S_0pS_{0h}pS_{0h2}p$; $N_0pN_{0l}pN_{0l2}p$;
label set
S_0ws_r; S_0ps_r; S_0ws_l; S_0ps_l; N_0ws_l; N_0ps_l;

- d distance
- v_l/v_r valency (related to the number of dependents)
- l dependency label
- s_l/s_r labelset

Recap: about linear combination

$$p(y|x;\theta) = \frac{\exp(\theta^\top f(x,y))}{\sum_{y' \in \mathcal{Y}} \exp(\theta^\top f(x,y'))}$$

$\cdots$	0	0	1	0	1	0	0	1	0	1	0	0	0	0	0	0	$\cdots$

f_{1001}: if word$_{-2}$=*some* and tag=N

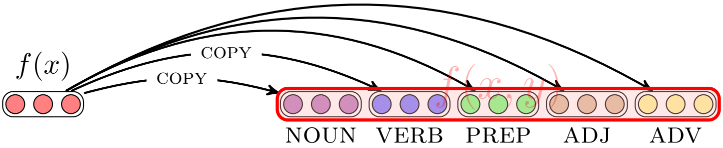
is θ_{1001} positive large?
vote for yes

Questions

Can we automate the design of features?

Is linear combination justified?

Learning feature representations



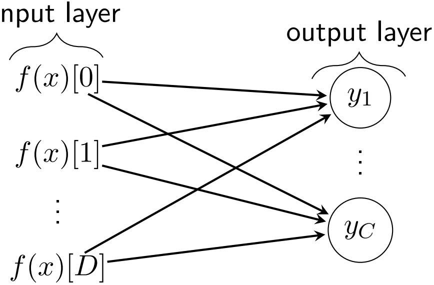
$$p(y|x;\theta) = \frac{\exp(\theta^\top f(x,y))}{\sum_{y'\in\mathcal{Y}} \exp(\theta^\top f(x,y'))} \longrightarrow \frac{\exp(\theta_y^\top f(x)))}{\sum_{y'\in\mathcal{Y}} \exp(\theta_{y'}^\top f(x))}$$

automatically induce f — f_θ

Feedforward Neural Networks

$$p(y|x;\theta) = \frac{\exp(\theta^\top f(x,y))}{\sum_{y' \in \mathcal{Y}} \exp(\theta^\top f(x,y'))} \longrightarrow \frac{\exp(\theta_y^\top f(x))}{\sum_{y' \in \mathcal{Y}} \exp(\theta_{y'}^\top f(x))}$$

As a simple neural network



$$\boxed{\texttt{softmax}(Wx)}$$

`softmax`: $\mathbb{R}^k \to \mathbb{R}^k$

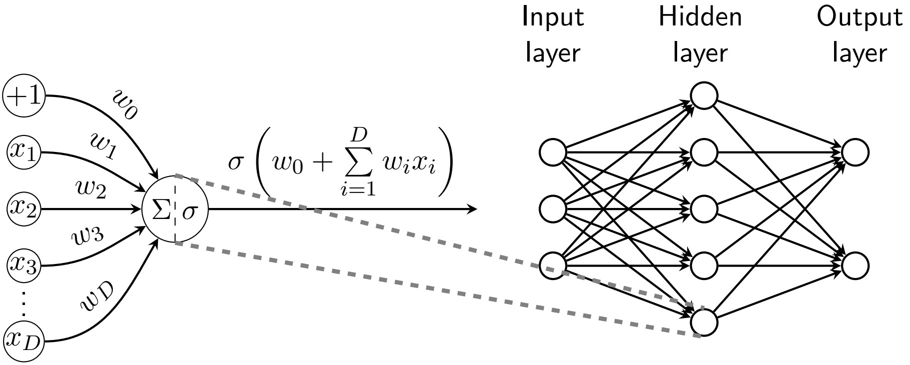
$$\langle x_1, x_2, \ldots, x_k \rangle \to \left\langle \frac{e^{x_1}}{\sum_j^k e^{x_j}}, \frac{e^{x_2}}{\sum_j^k e^{x_j}}, \ldots, \frac{e^{x_k}}{\sum_j^k e^{x_j}} \right\rangle$$

Sigmoid σ: $\mathbb{R} \to \mathbb{R}$

$$x \to \frac{1}{1+e^{-x}}$$

Elementwise σ: $\mathbb{R}^k \to \mathbb{R}^k$

$$\langle x_1, \ldots, x_k \rangle \to \left\langle \frac{1}{1+e^{-x_1}}, \ldots, \frac{1}{1+e^{-x_k}} \right\rangle$$



Neural network is a nesting of 'functions'

- 2-layers: $f = \sigma(W_2\sigma(W_1 x))$
- 3-layers: $f = \sigma(W_3\sigma(W_2\sigma(W_1 x)))$
- 4-layers: $f = \sigma(W_4\sigma(W_3\sigma(W_2\sigma(W_1 x))))$
- ...

$$h = g(w^\top x + b)$$

The activation function $g(z)$ should be non-linear

- The rectified linear unit function

$$\mathsf{ReLU}(z) = \max(0, z)$$

- The hyperbolic tangent function

$$\tanh(z) = \frac{e^{2z} - 1}{e^{2z} + 1}$$

How to obtain the model?

- Assume there is a good annotated corpus

$$\left\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(l)}, y^{(l)})\right\}$$

- Assume there is a loss function $l(y, \hat{y})$ as a way to describe how close a prediction is to the corresponding target!
- Cross-entropy for multi-class classification

$$l(y, \hat{y}; \theta) = -\sum_{i=1}^{l} \sum_{k=1}^{K} (y_k^{(i)} \cdot \log \hat{y}_k^{(i)})$$

- Find optimal weights W using stochastic gradient descent, such that the loss function is minimized
    - Compute gradients with backpropagation
    - Iterate many times over training set

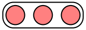
Advancing Dependency Parsing

From feature template to feature vector

distance
S_0wd; S_0pd; N_0wd; N_0pd; S_0wN_0wd; S_0pN_0pd;
valency
S_0wv_r; S_0pv_r; S_0wv_l; S_0pv_l; N_0wv_l; N_0pv_l;
unigrams
$S_{0h}w$; $S_{0h}p$; S_0l; $S_{0l}w$; $S_{0l}p$; $S_{0l}l$; $S_{0r}w$; $S_{0r}p$; $S_{0r}l$; $N_{0l}w$; $N_{0l}p$; $N_{0l}l$;
third-order
$S_{0h2}w$; $S_{0h2}p$; $S_{0h}l$; $S_{0l2}w$; $S_{0l2}p$; $S_{0l2}l$; $S_{0r2}w$; $S_{0r2}p$; $S_{0r2}l$; $N_{0l2}w$; $N_{0l2}p$; $N_{0l2}l$; $S_0pS_{0l}pS_{0l2}p$; $S_0pS_{0r}pS_{0r2}p$; $S_0pS_{0h}pS_{0h2}p$; $N_0pN_{0l}pN_{0l2}p$;
label set
S_0ws_r; S_0ps_r; S_0ws_l; S_0ps_l; N_0ws_l; N_0ps_l;

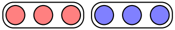
From feature template to feature vector

distance
S_0wd; S_0pd; N_0wd; N_0pd; S_0wN_0wd; S_0pN_0pd;
valency
S_0wv_r; S_0pv_r; S_0wv_l; S_0pv_l; N_0wv_l; N_0pv_l;
unigrams
$S_{0h}w$; $S_{0h}p$; S_0l; $S_{0l}w$; $S_{0l}p$; $S_{0l}l$; $S_{0r}w$; $S_{0r}p$; $S_{0r}l$; $N_{0l}w$; $N_{0l}p$; $N_{0l}l$;
third-order
$S_{0h2}w$; $S_{0h2}p$; $S_{0h}l$; $S_{0l2}w$; $S_{0l2}p$; $S_{0l2}l$; $S_{0r2}w$; $S_{0r2}p$; $S_{0r2}l$; $N_{0l2}w$; $N_{0l2}p$; $N_{0l2}l$; $S_0pS_{0l}pS_{0l2}p$; $S_0pS_{0r}pS_{0r2}p$; $S_0pS_{0h}pS_{0h2}p$; $N_0pN_{0l}pN_{0l2}p$;
label set
S_0ws_r; S_0ps_r; S_0ws_l; S_0ps_l; N_0ws_l; N_0ps_l;



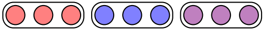
From feature template to feature vector

distance
S_0wd; S_0pd; N_0wd; N_0pd; S_0wN_0wd; S_0pN_0pd;
valency
S_0wv_r; S_0pv_r; S_0wv_l; S_0pv_l; N_0wv_l; N_0pv_l;
unigrams
$S_{0h}w$; $S_{0h}p$; S_0l; $S_{0l}w$; $S_{0l}p$; $S_{0l}l$; $S_{0r}w$; $S_{0r}p$; $S_{0r}l$; $N_{0l}w$; $N_{0l}p$; $N_{0l}l$;
third-order
$S_{0h2}w$; $S_{0h2}p$; $S_{0h}l$; $S_{0l2}w$; $S_{0l2}p$; $S_{0l2}l$; $S_{0r2}w$; $S_{0r2}p$; $S_{0r2}l$; $N_{0l2}w$; $N_{0l2}p$; $N_{0l2}l$; $S_0pS_{0l}pS_{0l2}p$; $S_0pS_{0r}pS_{0r2}p$; $S_0pS_{0h}pS_{0h2}p$; $N_0pN_{0l}pN_{0l2}p$;
label set
S_0ws_r; S_0ps_r; S_0ws_l; S_0ps_l; N_0ws_l; N_0ps_l;



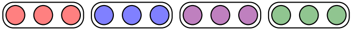
From feature template to feature vector

distance
S_0wd; S_0pd; N_0wd; N_0pd; S_0wN_0wd; S_0pN_0pd;
valency
S_0wv_r; S_0pv_r; S_0wv_l; S_0pv_l; N_0wv_l; N_0pv_l;
unigrams
$S_{0h}w$; $S_{0h}p$; S_0l; $S_{0l}w$; $S_{0l}p$; $S_{0l}l$; $S_{0r}w$; $S_{0r}p$; $S_{0r}l$; $N_{0l}w$; $N_{0l}p$; $N_{0l}l$;
third-order
$S_{0h2}w$; $S_{0h2}p$; $S_{0h}l$; $S_{0l2}w$; $S_{0l2}p$; $S_{0l2}l$; $S_{0r2}w$; $S_{0r2}p$; $S_{0r2}l$; $N_{0l2}w$; $N_{0l2}p$; $N_{0l2}l$; $S_0pS_{0l}pS_{0l2}p$; $S_0pS_{0r}pS_{0r2}p$; $S_0pS_{0h}pS_{0h2}p$; $N_0pN_{0l}pN_{0l2}p$;
label set
S_0ws_r; S_0ps_r; S_0ws_l; S_0ps_l; N_0ws_l; N_0ps_l;



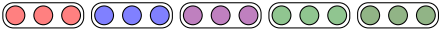
From feature template to feature vector

distance
S_0wd; S_0pd; N_0wd; N_0pd; S_0wN_0wd; S_0pN_0pd;
valency
S_0wv_r; S_0pv_r; S_0wv_l; S_0pv_l; N_0wv_l; N_0pv_l;
unigrams
$S_{0h}w$; $S_{0h}p$; S_0l; $S_{0l}w$; $S_{0l}p$; $S_{0l}l$; $S_{0r}w$; $S_{0r}p$; $S_{0r}l$; $N_{0l}w$; $N_{0l}p$; $N_{0l}l$;
third-order
$S_{0h2}w$; $S_{0h2}p$; $S_{0h}l$; $S_{0l2}w$; $S_{0l2}p$; $S_{0l2}l$; $S_{0r2}w$; $S_{0r2}p$; $S_{0r2}l$; $N_{0l2}w$; $N_{0l2}p$; $N_{0l2}l$; $S_0pS_{0l}pS_{0l2}p$; $S_0pS_{0r}pS_{0r2}p$; $S_0pS_{0h}pS_{0h2}p$; $N_0pN_{0l}pN_{0l2}p$;
label set
S_0ws_r; S_0ps_r; S_0ws_l; S_0ps_l; N_0ws_l; N_0ps_l;



From feature template to feature vector

distance
S_0wd; S_0pd; N_0wd; N_0pd; S_0wN_0wd; S_0pN_0pd;
valency
S_0wv_r; S_0pv_r; S_0wv_l; S_0pv_l; N_0wv_l; N_0pv_l;
unigrams
$S_{0h}w$; $S_{0h}p$; S_0l; $S_{0l}w$; $S_{0l}p$; $S_{0l}l$; $S_{0r}w$; $S_{0r}p$; $S_{0r}l$; $N_{0l}w$; $N_{0l}p$; $N_{0l}l$;
third-order
$S_{0h2}w$; $S_{0h2}p$; $S_{0h}l$; $S_{0l2}w$; $S_{0l2}p$; $S_{0l2}l$; $S_{0r2}w$; $S_{0r2}p$; $S_{0r2}l$; $N_{0l2}w$; $N_{0l2}p$; $N_{0l2}l$; $S_0pS_{0l}pS_{0l2}p$; $S_0pS_{0r}pS_{0r2}p$; $S_0pS_{0h}pS_{0h2}p$; $N_0pN_{0l}pN_{0l2}p$;
label set
S_0ws_r; S_0ps_r; S_0ws_l; S_0ps_l; N_0ws_l; N_0ps_l;

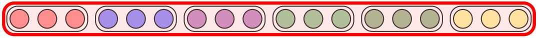


From feature template to feature vector

distance
S_0wd; S_0pd; N_0wd; N_0pd; S_0wN_0wd; S_0pN_0pd;
valency
S_0wv_r; S_0pv_r; S_0wv_l; S_0pv_l; N_0wv_l; N_0pv_l;
unigrams
$S_{0h}w$; $S_{0h}p$; S_0l; $S_{0l}w$; $S_{0l}p$; $S_{0l}l$; $S_{0r}w$; $S_{0r}p$; $S_{0r}l$; $N_{0l}w$; $N_{0l}p$; $N_{0l}l$;
third-order
$S_{0h2}w$; $S_{0h2}p$; $S_{0h}l$; $S_{0l2}w$; $S_{0l2}p$; $S_{0l2}l$; $S_{0r2}w$; $S_{0r2}p$; $S_{0r2}l$; $N_{0l2}w$; $N_{0l2}p$; $N_{0l2}l$; $S_0pS_{0l}pS_{0l2}p$; $S_0pS_{0r}pS_{0r2}p$; $S_0pS_{0h}pS_{0h2}p$; $N_0pN_{0l}pN_{0l2}p$;
label set
S_0ws_r; S_0ps_r; S_0ws_l; S_0ps_l; N_0ws_l; N_0ps_l;

From feature template to feature vector

distance
S_0wd; S_0pd; N_0wd; N_0pd; S_0wN_0wd; S_0pN_0pd;
valency
S_0wv_r; S_0pv_r; S_0wv_l; S_0pv_l; N_0wv_l; N_0pv_l;
unigrams
$S_{0h}w$; $S_{0h}p$; S_0l; $S_{0l}w$; $S_{0l}p$; $S_{0l}l$; $S_{0r}w$; $S_{0r}p$; $S_{0r}l$; $N_{0l}w$; $N_{0l}p$; $N_{0l}l$;
third-order
$S_{0h2}w$; $S_{0h2}p$; $S_{0h}l$; $S_{0l2}w$; $S_{0l2}p$; $S_{0l2}l$; $S_{0r2}w$; $S_{0r2}p$; $S_{0r2}l$; $N_{0l2}w$; $N_{0l2}p$; $N_{0l2}l$; $S_0pS_{0l}pS_{0l2}p$; $S_0pS_{0r}pS_{0r2}p$; $S_0pS_{0h}pS_{0h2}p$; $N_0pN_{0l}pN_{0l2}p$;
label set
S_0ws_r; S_0ps_r; S_0ws_l; S_0ps_l; N_0ws_l; N_0ps_l;



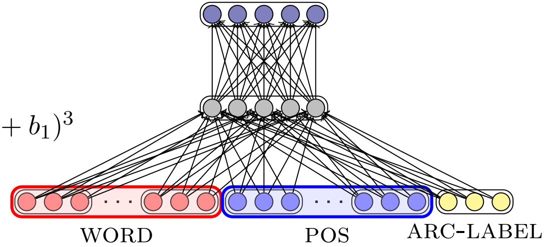
word representation x^w

Neural transition-based parsing (Chen and Manning 2014)

softmax layer
$p = \texttt{softmax}(W_2 h)$

hidden layer $g = (\cdot)^3$
$h = (W_1^w x^w + W_1^t x^t + W_1^l x^l + b_1)^3$

Input layer: $[x^w, x^t, x^l]$



Cube activation function

Using $g(x) = x^3$ can model the product terms of x_i, x_j, x_k for any three different elements at the input layer directly.

Neural transition-based parsing (Chen and Manning 2014)

English parsing to Stanford Dependencies

- Unlabeled attachment score (UAS): #{correct head}/#{total head}
- Labeled attachment score (LAS): #{correct head with correct label}/#{total head}

Parser	UAS	LAS	sent./s
MaltParser	89.8	87.2	469
MSTParser	91.4	88.1	10
TurboParser	92.3	89.6	8
C & M 2014	92.0	89.7	654

- The first simple, successful neural dependency parser
- The dense representations let it outperform other greedy parsers in both accuracy and speed
- Neural networks can accurately determine the structure of sentences, supporting interpretation

from Stanford CS224N/Ling284

Some General Comments on Neural NLP

Idea: Deep learning simplifies machine learning

- Why has deep learning taken over NLP?
- Deep learning simplifies the design of probabilistic models, by replacing complex dependencies and independence assumptions with universal function approximators.
- Deep learning gives us representation learning: data representations are learned rather than engineered.
- Learned representations are easy to obtain and reusable, enabling multi-task learning.
- Deep learning provides a uniform, flexible, trainable framework that can easily mix and match different data types: strings, labels, trees, graphs, data, and images.

In short: deep learning solves the difficulties of applying machine learning to NLP... it does not solve NLP, which is still difficult!

from A Lopez' slide

Observations of NNLP (so far): positives

- Really important change in state-of-the-art for many applications: e.g., language models for speech. Now the default approach for many tasks.
- Multi-modal experiments are now much more feasible.
- Models are learning structure with out hand-crafting of features.
- Structure learned for one task (e.g.,prediction) applicable to others with limited training data.
- Lots of toolkits etc
- Huge space of new models, far more research going onin NLP, far more industrial research . . .

Observations of NNLP (so far): negatives

- Models are made as powerful as possible to the point they are “barely possible to train or use” (`http://www.deeplearningbook.org` 16.7).
- Tuning hyperparameters is a matter of much experimentation.
- Statistical validity of results often questionable.
- Many myths, massive hype and almost no publication of negative results: but there are some NLP tasks where deep learning is not giving much improvement in results.
- Weird results: e.g., ‘33rpm’ normalized to ‘thirty two revolutions per minute’
  `https://arxiv.org/ftp/arxiv/papers/1611/1611.00068.pdf`
- Adversarial examples

New methodology required for NLP?

- Perspective here is applied machine learning, e.g. Collobert et al (2011)
  natural language processing from scratch
- Methodological issues are fundamental to deep learning: e.g., subtle biases in training data will be picked up.
- Old tasks and old data possibly no longer appropriate.
- The lack of predefined interpretation of the latent variables is what makes the models more flexible/powerful ...
- but the models are usually not interpretable by humans after training: serious practical and ethical issues.

Readings

- D Chen and C Manning. 2014. A Fast and Accurate Dependency Parser using Neural Networks. `www.aclweb.org/anthology/D14-1082/`