Deep Learning for Language Understanding
Step 1: Word Vectors

Christopher Manning

Stanford University

@chrmanning ❀ @stanfordnlp

Harker Programming Invitational 2017

What is Machine Learning?

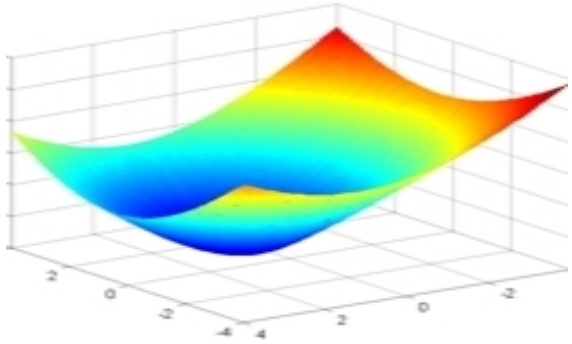
Machine learning is the approach where, instead of programming computers to follow instructions, we program them to learn to do things

However, most machine learning methods work well because of carefully **human-designed representations** and **input features**

- E.g., features for finding named entities like person or organization names

Previous word	*at*
Current word	*Grace*
Beginning bigram	*<G*
Prev and curr POS	IN NNP
Previous state	Other
Current signature	Xx
Prev state, cur sig	O-Xx
Prev-cur-next sig	x-Xx-Xx
P. state - p-cur sig	O-x-Xx
…	

Machine learning becomes just optimizing weights to best make a final prediction

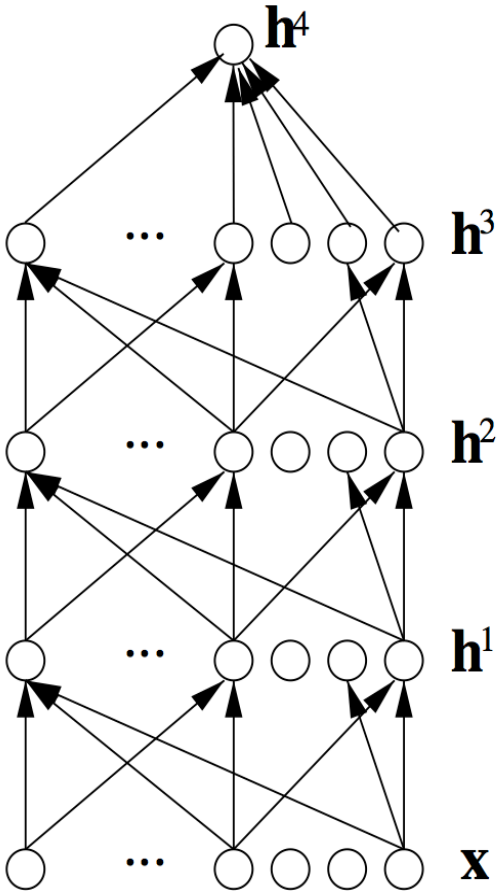


What is Deep Learning?

Representation learning is a subfield of machine learning, where we attempt to automatically learn the good features or representations

Deep learning algorithms do this by attempting to learn multiple levels of representation **h** and an output

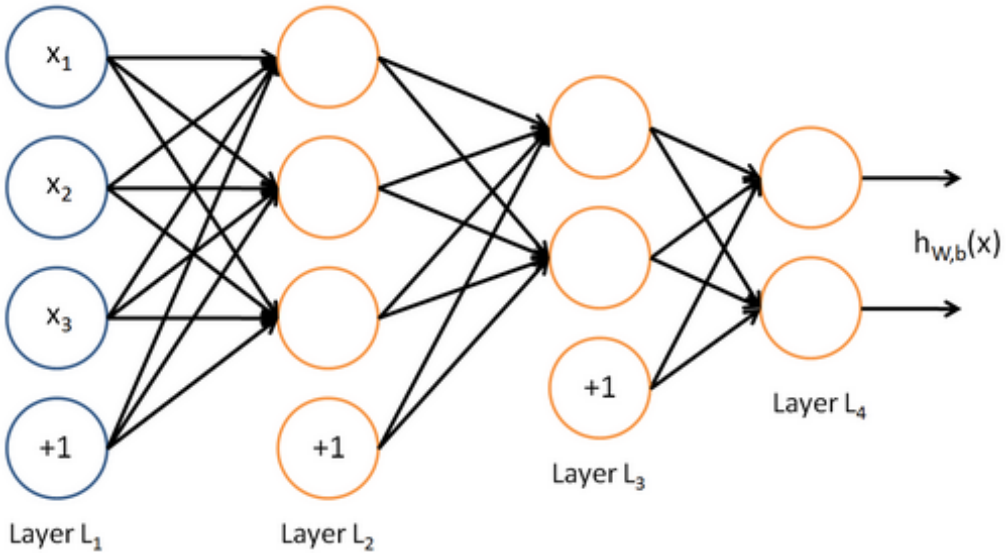
From "raw" inputs **x**
(e.g., sound, characters, or words)



What is a Neural Network?

(Artificial) Neural Networks work by using distributed representations of concepts as vectors of real numbers

They compute representations by matrix multiplies from one layer to another (followed by an element-wise rescaling)



What is Computational Linguistics/NLP?

Computational linguistics or natural language processing is a field at the intersection of

- artificial intelligence/computer science
- and linguistics (the science of human languages)

Goal: for computers to process or understand human languages in order to perform tasks that are useful, e.g.,

- An agent that can make appointments or order things
- Question answering
- Machine translation

E.g., Siri, Google Assistant, Cortana, … thank you, mobile!!!

Commercial world

amazon

the problem session for your course, CS224n in Gates B03 from
1:15-2:05
Skilling Aud
Create New iCal Event...
Show This Date in iCal...

PEARSON
Knowledge Technologies

Google™

ski resort

SDL
Because Business is Global

YAHOO!

Microsoft®

Sugar Bowl **Ski** Lodging
Escape to our Snowbound Village.
Fresh tracks steps from your room.
www.sugarbowl.com

BETA
hakia®
search for meaning

Their are many approaches,
out

collective intellect

J.D. POWER
AND ASSOCIATES®

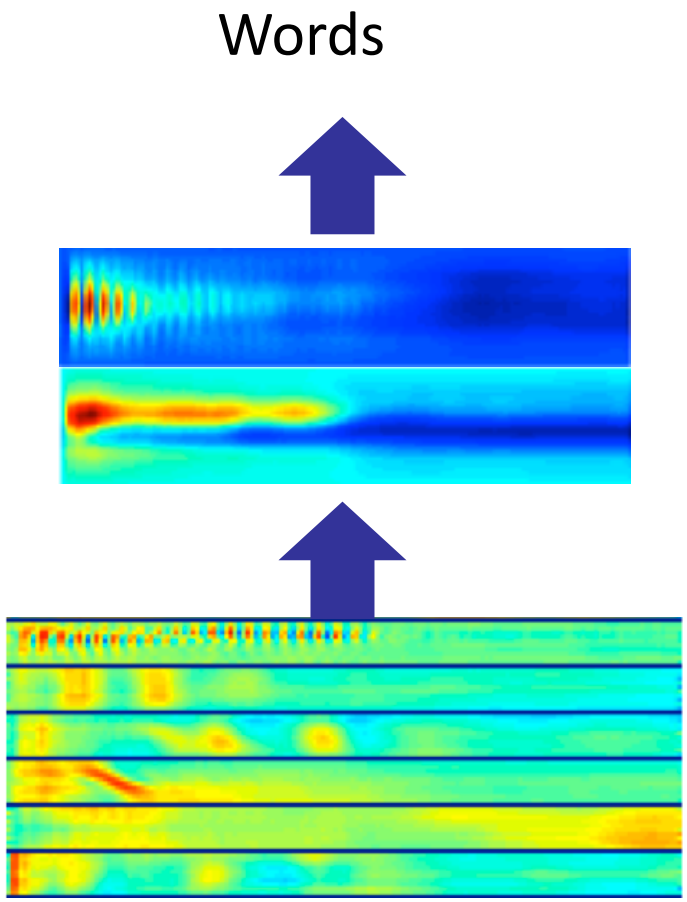
Nielsen
BuzzMetrics

Deep Learning for Speech

The first breakthrough results of deep learning on a large dataset happened in speech recognition

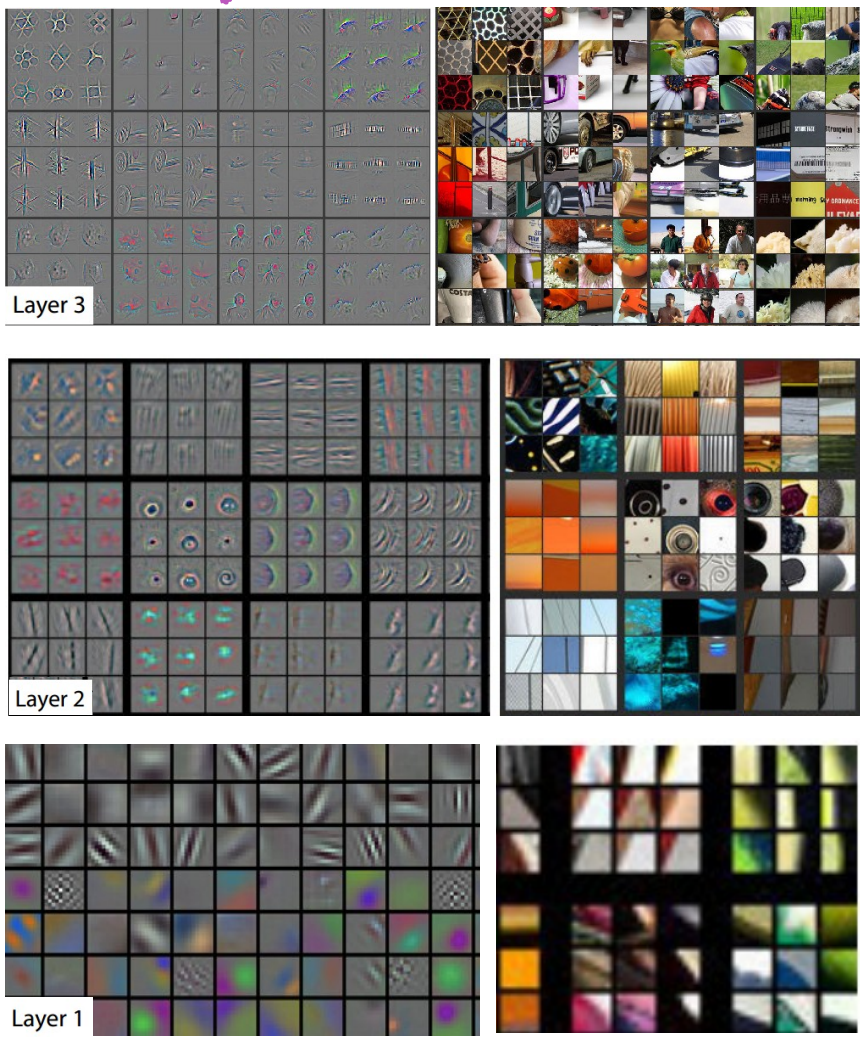
- Context-Dependent Pre-trained Deep Neural Networks for Large Vocabulary Speech Recognition (Dahl et al. 2010)

Acoustic model	Recog WER	RT03S FSH	Hub5 SWB
Traditional GMM features	1-pass −adapt	**27.4**	**23.6**
Deep Learning	1-pass −adapt	**18.5** (−33%)	**16.1** (−32%)



Deep Learning for Computer Vision

The breakthrough DL paper: ImageNet Classification with Deep Convolutional Neural Networks by Krizhevsky, Sutskever, & Hinton, 2012, U. Toronto. 37% error red.



Zeiler and Fergus (2013)

Word vectors: From symbolic to distributed word representations

The vast majority of rule-based and statistical natural language processing or web search work regarded words as atomic symbols: hotel, conference

In machine learning vector space terms, this is a vector with one 1 and a lot of zeroes

[0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]

Deep learning people call this a "one-hot" representation

From symbolic to distributed word representations

Its problem, e.g., for web search:

- If user searches for [Dell notebook battery size], we would like to match documents with "Dell laptop battery capacity"
- If user searches for [Seattle motel], we would like to match documents containing "Seattle hotel"

But

motel [0 0 0 0 0 0 0 0 0 0 1 0 0 0 0]^T
hotel [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0] = 0

Our query and document vectors are **orthogonal**

There is no natural notion of similarity in a set of one-hot vectors

A solution via distributional similarity-based representations

You can get a lot of value by representing a word by means of its neighbors

"You shall know a word by the company it keeps"

(J. R. Firth 1957: 11)

One of the most successful ideas of modern NLP

government debt problems turning into	banking	crises as has happened in
saying that Europe needs unified	banking	regulation to replace the hodgepodge

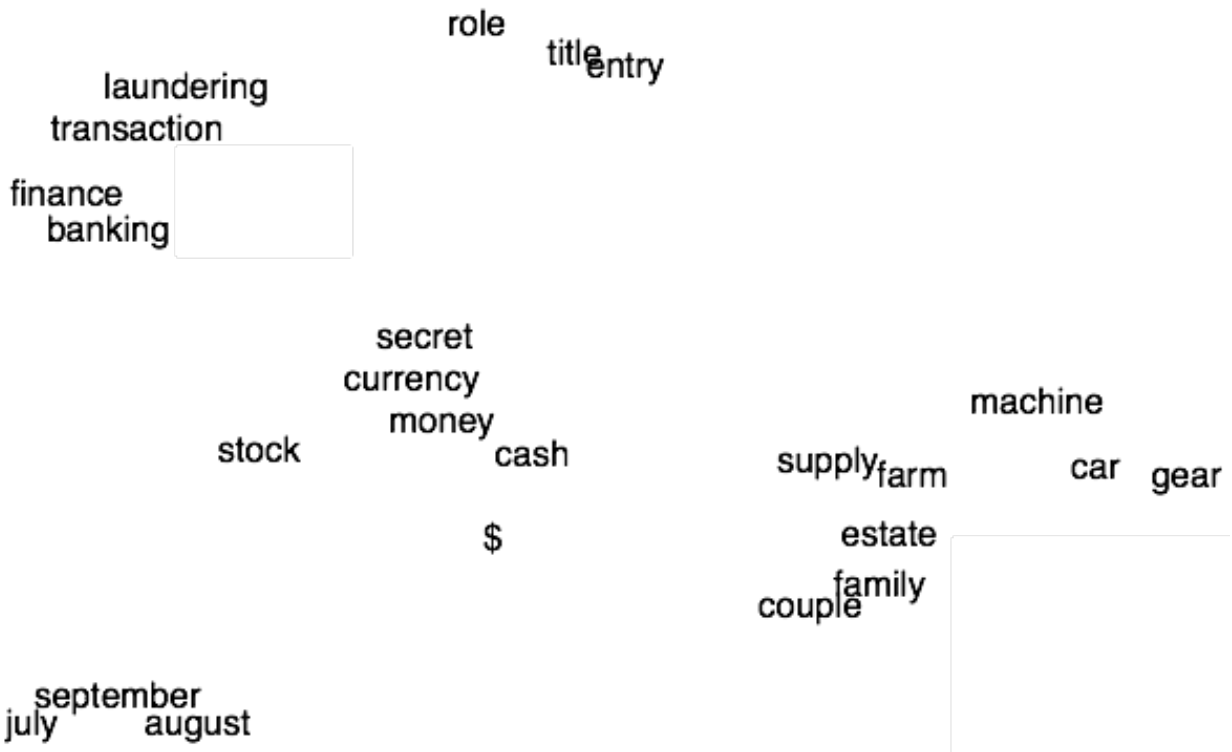
↖ These words will represent *banking* ↗

Word meaning as a vector

We build a dense vector for each word type, chosen so that it is good at predicting other words appearing in its context

… those other words also being represented by vectors

$$\textit{currency} = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}$$



Basic idea of learning neural network word embeddings

We define a model that aims to predict between a center word w_t and context words in terms of word vectors

$$p(\text{context} \mid w_t) = \ldots$$

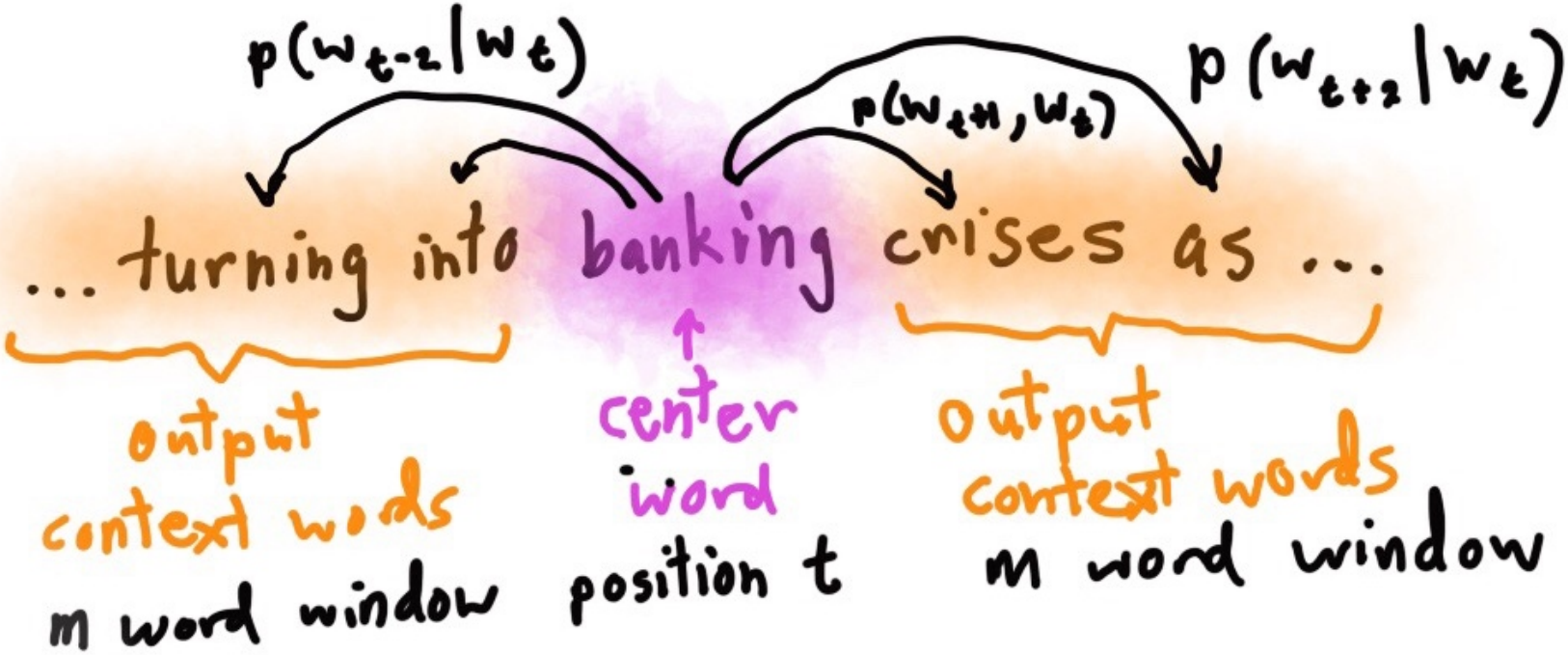
which has a loss function, e.g.,

$$J = 1 - p(w_{-t} \mid w_t)$$

We look at many positions t in a big amount of text

We keep adjusting the vector representations of words to minimize this loss

Skip-gram prediction



Details of Word2Vec

Predict surrounding words in a window of radius *m* of every word

For $p(w_{t+j}|w_t)$ the simplest first formulation is

$$p(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w=1}^{V} \exp(u_w^T v_c)}$$

where *o* is the outside (or output) word index, *c* is the center word index, v_c and u_o are "center" and "outside" vectors of indices *c* and *o*

Softmax using word *c* to obtain probability of word *o*

Dot products

Dot product

$$u^T v = u \cdot v = \sum_{i=1}^{n} u_i v_i$$

Bigger if u and v are more similar!

Iterate over $w = 1 \ldots W$: $u_w^T v$ means:

Work out how similar each word is to v!

$$p(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w=1}^{V} \exp(u_w^T v_c)}$$

Softmax function: Standard map from $\mathbb{R}^V$ to a probability distribution

Softmax

Exponentiate to make positive

Normalize to give probability

$$p_i = \frac{e^{u_i}}{\sum_j e^{u_i}}$$

To learn good word vectors: Compute all vector gradients!

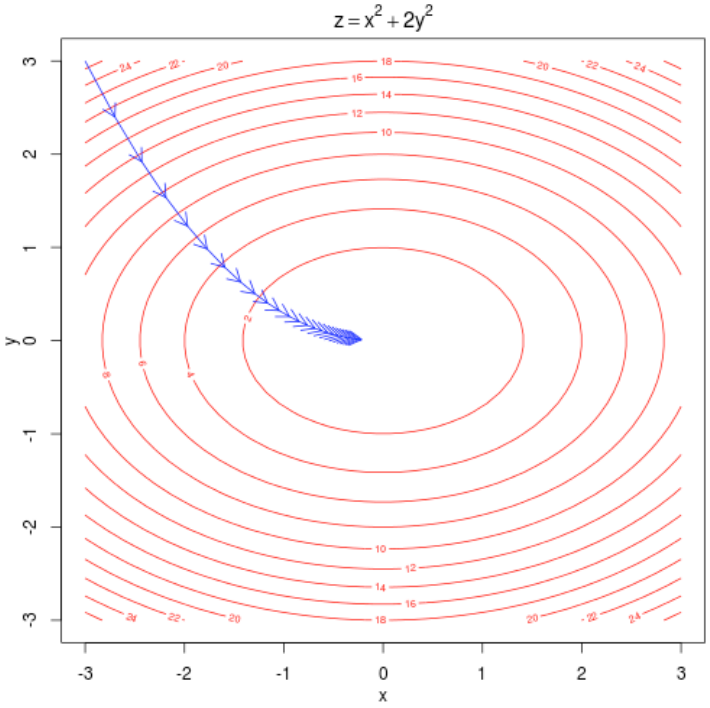
- We define the set of **all** parameters of the model in terms of one long vector θ
- In our case with d-dimensional vectors (perhaps $d = 300$), and V many words:

$$\theta = \begin{bmatrix} v_{aardvark} \\ v_a \\ \vdots \\ v_{zebra} \\ u_{aardvark} \\ u_a \\ \vdots \\ u_{zebra} \end{bmatrix} \in \mathbb{R}^{2dV}$$

- We then want to "optimize" these parameters

Intuition of how to minimize loss for a simple function over two parameters

We start at a random point and walk in the steepest direction, which is given by the derivative of the function

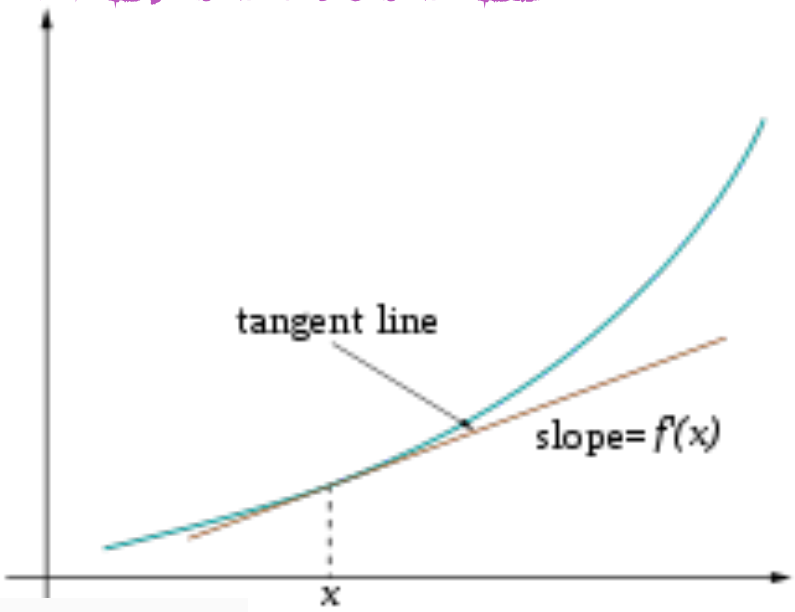


Contour lines show points of equal value of objective function

Descending by using derivatives

We will minimize a cost function by gradient descent

Trivial example: (from Wikipedia)
Find a local minimum of the function
$f(x) = x^4 - 3x^3 + 2$,
with derivative $f'(x) = 4x^3 - 9x^2$




```
x_old = 0
x_new = 6 # The algorithm starts at x=6
eps = 0.01 # step size
precision = 0.00001

def f_derivative(x):
    return 4 * x**3 - 9 * x**2

while abs(x_new - x_old) > precision:
    x_old = x_new
    x_new = x_old - eps * f_derivative(x_old)

print("Local minimum occurs at", x_new)
```

Subtracting a fraction of the gradient moves you towards the minimum!

Vanilla Gradient Descent Code

$$\theta^{new} = \theta^{old} - \alpha \nabla_\theta J(\theta)$$

```
while True:
    theta_grad = evaluate_gradient(J,corpus,theta)
    theta = theta - alpha * theta_grad
```

Training/optimizing a neural network is really all the chain rule!

Chain rule! If $y = f(u)$ and $u = g(x)$, i.e. $y = f(g(x))$, then:

$$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx} = \frac{df(u)}{du}\frac{dg(x)}{dx}$$

Simple example: $\frac{dy}{dx} = \frac{d}{dx}5(x^3+7)^4$

$$y = f(u) = 5u^4 \qquad u = g(x) = x^3 + 7$$

$$\frac{dy}{du} = 20u^3 \qquad \frac{du}{dx} = 3x^2$$

$$\frac{dy}{dx} = 20(x^3+7)^3 \cdot 3x^2$$

Objective Function

Maximize $$J'(\theta) = \prod_{t=1}^{T} \prod_{\substack{-m \le j \le m \\ j \neq 0}} p(w'_{t+j} | w_t ; \theta)$$

Or minimize neg. log likelihood $$J(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \neq 0}} \log p(w'_{t+j} | w_t)$$

[negate to minimize; log is monotone]

text length

window size

where $$p(o|c) = \frac{\exp(u_o^T v_c)}{\sum_{w=1}^{V} \exp(u_w^T v_c)}$$

word IDs

We now take derivatives to work out minimum

Each word type (vocab entry) has two word representations: as center word and context word

$$\frac{\partial}{\partial v_c} \log \frac{\exp(u_o^T v_c)}{\sum_{w=1}^{V} \exp(u_w v_c)}$$

$$= \underbrace{\frac{\partial}{\partial v_c} \log \exp(u_o^T v_c)}_{①} - \underbrace{\frac{\partial}{\partial v_c} \log \sum_{w=1}^{V} \exp(u_w^T v_c)}_{②}$$

① $\frac{\partial}{\partial v_c} \log \exp(u_o^T v_c) = \frac{\partial}{\partial v_c} u_o^T v_c = u_o$

Vector! Not high school single variable calculus

inverses

You can do things one variable at a time, and this may be helpful when things get gnarly.

$$\forall j \quad \frac{\partial}{\partial (v_c)_j} u_o^T v_c = \frac{\partial}{\partial (v_c)_j} \sum_{i=1}^{d} (u_o)_i (v_c)_i$$

$$= (u_o)_j$$

Each term is zero except when $i = j$

② $\frac{\partial}{\partial v_c} \underbrace{\log}_{f} \underbrace{\sum_{w=1}^{V} \exp(u_w^T v_c)}_{z = g(v_c)}$

$= \underbrace{\frac{1}{\sum_{w=1}^{V} \exp(u_w^T v_c)}}_{\frac{\partial f}{\partial z}} \cdot \underbrace{\frac{\partial}{\partial v_c} \sum_{x=1}^{V} \exp(u_x^T v_c)}_{\text{Important to change index}}$

$\frac{\partial}{\partial v_c} f(\overbrace{g(v_c)}^{z}) = \frac{\partial f}{\partial z} \cdot \frac{\partial z}{\partial v_c}$ Use chain rule

$= \frac{1}{\sum_{w=1}^{V} \exp(u_w^T v_c)} \cdot \left(\sum_{x=1}^{V} \frac{\partial}{\partial v_c} \underbrace{\exp}_{f} \underbrace{(u_x^T v_c)}_{z = g(v_c)} \right)$ Move deriv inside sum

$\left(\sum_{x=1}^{V} \exp(u_x^T v_c) \frac{\partial}{\partial v_c} u_x^T v_c \right)$ Chain rule

$\left(\sum_{x=1}^{V} \exp(u_x^T v_c) \, u_x \right)$

$$\frac{\partial}{\partial v_c} \log\left(p(o|c)\right) = u_o - \frac{1}{\sum_{w=1}^{V} \exp(u_w^T v_c)} \cdot \left(\sum_{x=1}^{V} \exp(u_x^T v_c)\, u_x\right)$$

$$= u_o - \sum_{x=1}^{V} \frac{\exp(u_x^T v_c)}{\sum_{w=1}^{V} \exp(u_w^T v_c)}\, u_x$$

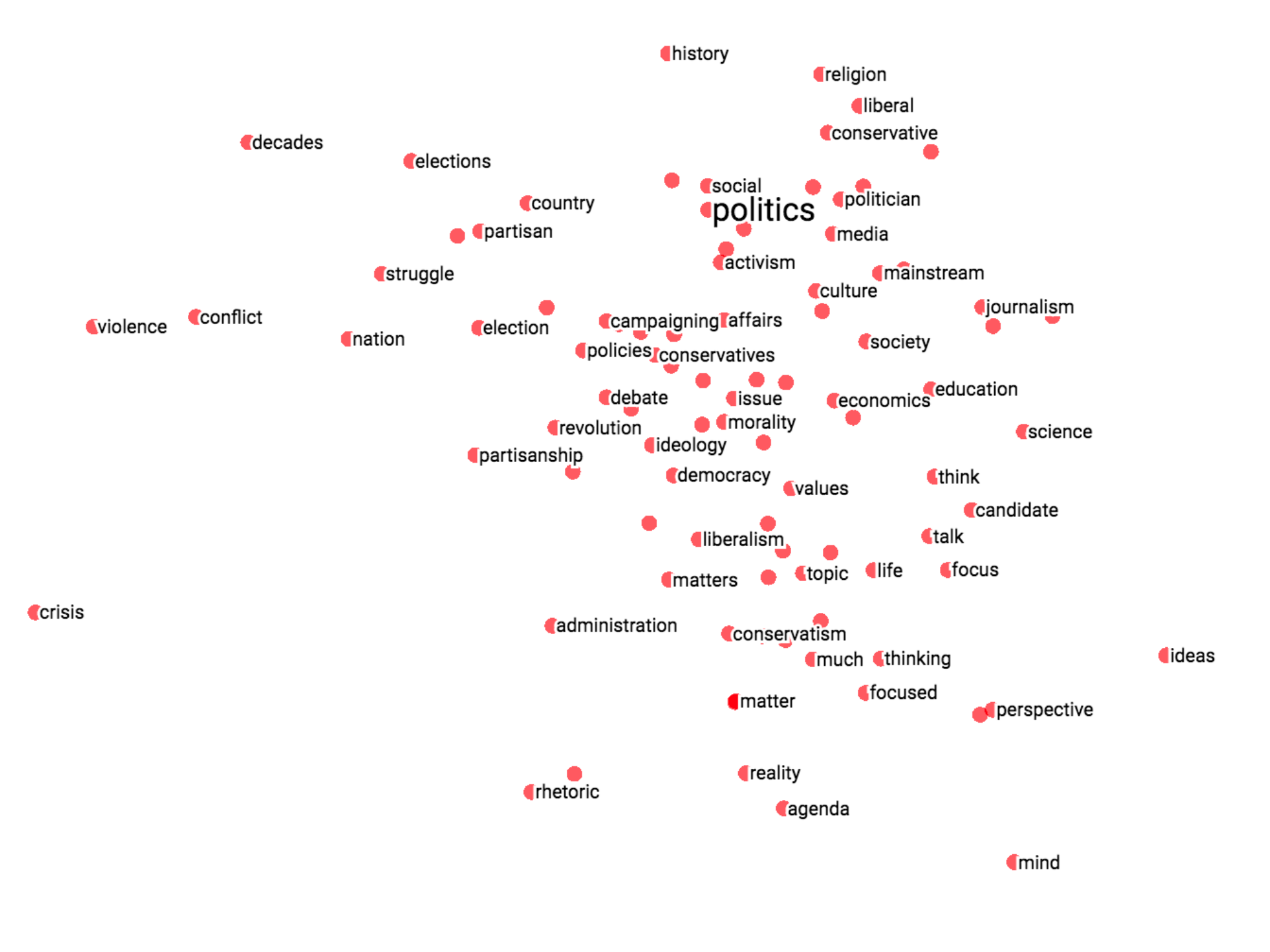
Distribute term across sum

$$= u_o - \sum_{x=1}^{V} p(x|c)\, u_x$$

This an expectation: average over all context vectors weighted by their probability

$$= \text{observed} - \text{expected}$$

This is just the derivatives for the center vector parameters
Also need derivatives for output vector parameters (they're similar)
Then we have derivative w.r.t. all parameters and can minimize


history
religion
liberal
conservative
decades
elections
social
politician
country
politics
partisan
media
activism
struggle
mainstream
culture
journalism
violence
conflict
election
campaigning
affairs
nation
society
policies
conservatives
debate
issue
economics
education
revolution
morality
science
ideology
partisanship
democracy
think
values
candidate
liberalism
talk
topic
life
focus
matters
crisis
administration
conservatism
much
thinking
ideas
matter
focused
perspective
reality
rhetoric
agenda
mind

Word similarities

Nearest words to frog:

1. frogs
2. toad
3. litoria
4. leptodactylidae
5. rana
6. lizard
7. eleutherodactylus

litoria

leptodactylidae

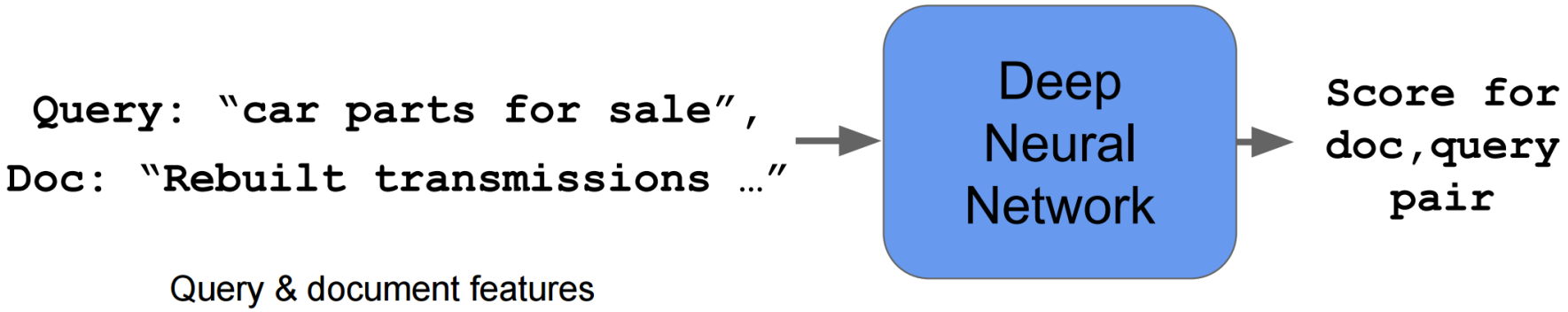
rana

eleutherodactylus

http://nlp.stanford.edu/projects/glove/

Distributed word representations can capture the long tail of web queries

Google's RankBrain

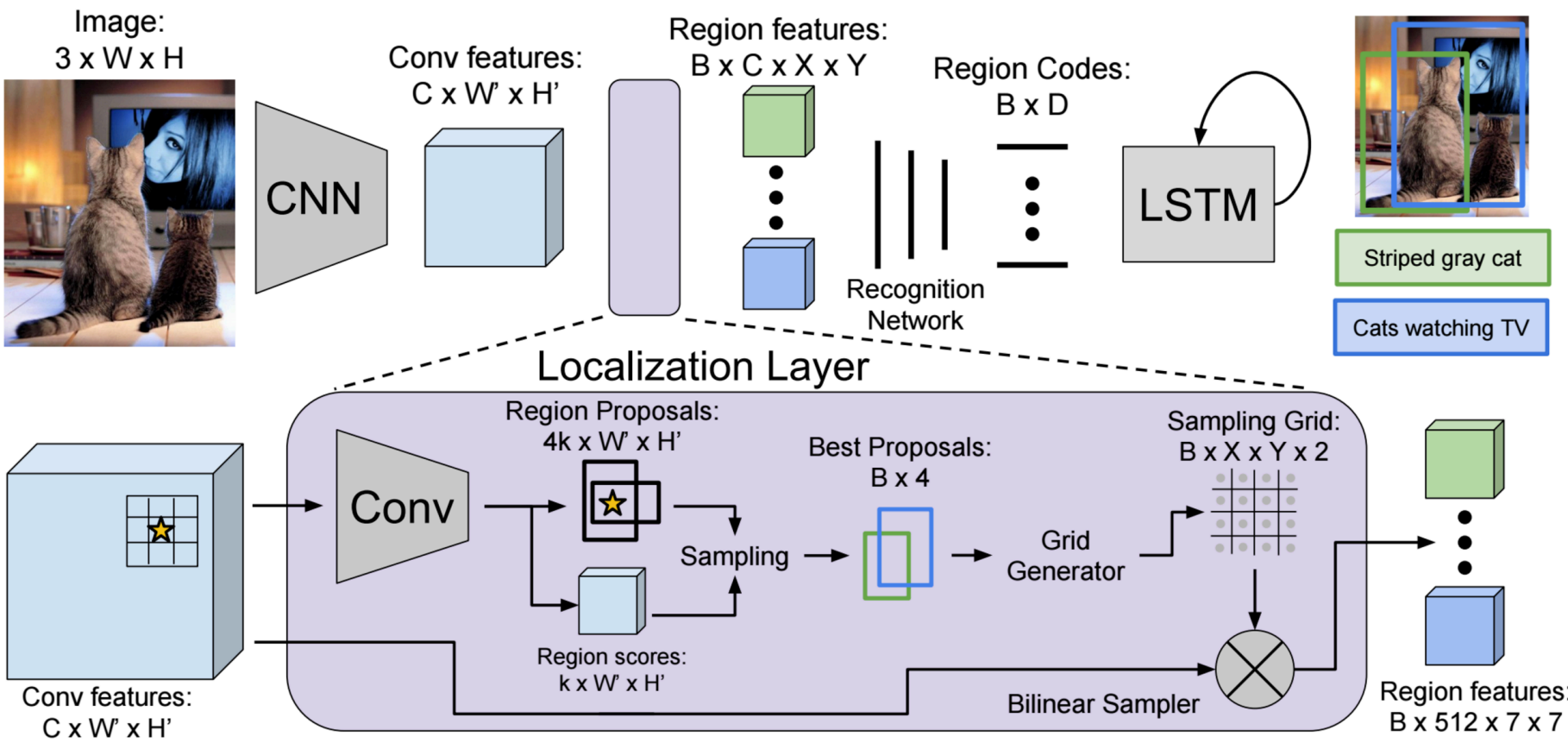


Not necessarily as good for very common queries

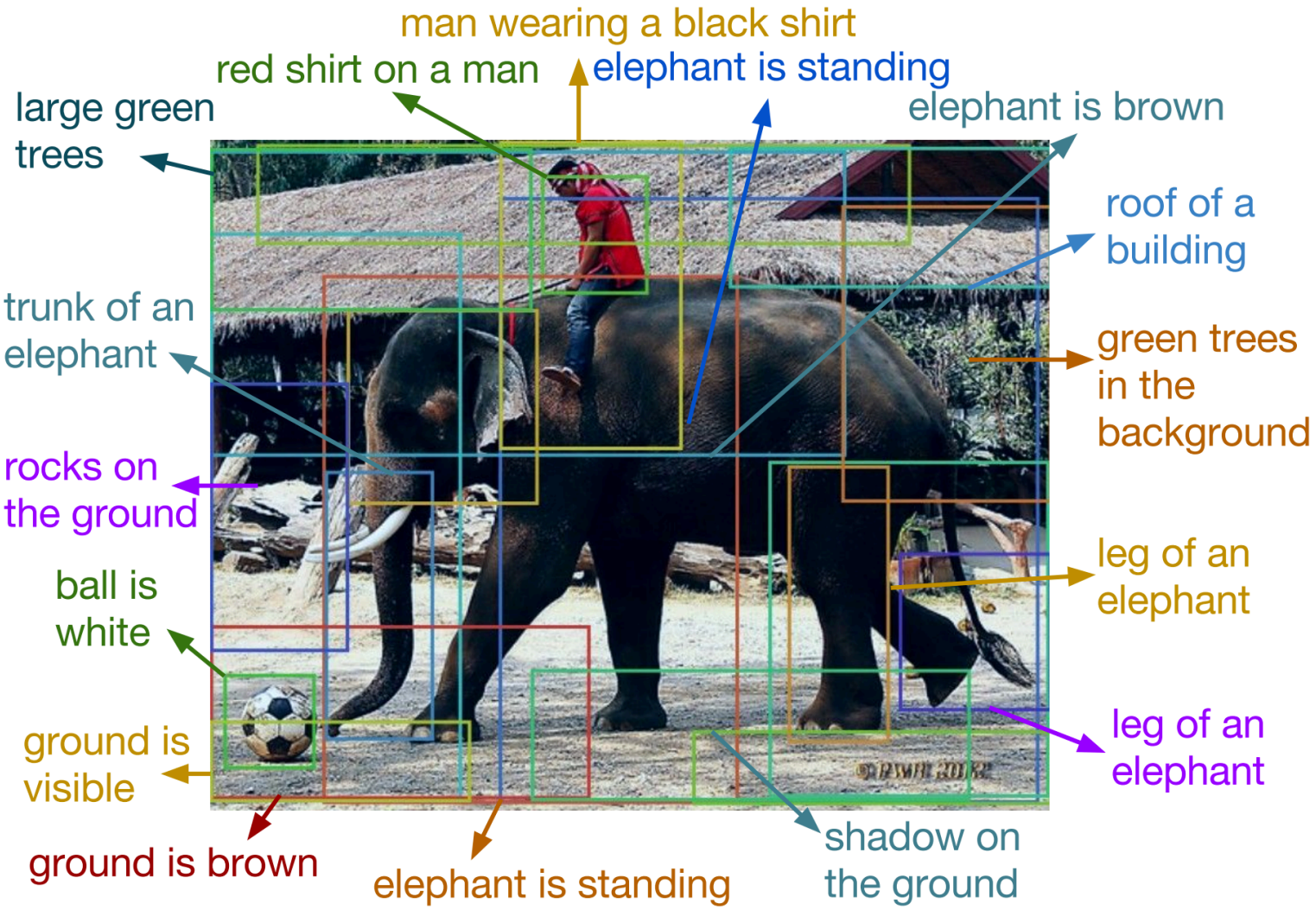
But great for seeing similarity in the tail

3rd most important ranking signal in Google web search!

Larger-scale deep learning systems
[Johnson, Karpathy, Fei-Fei 2015]



DenseCap: Fully Neural Network Localization and Text Generation



Summary

Actually understanding what people are saying with language – beyond just recognizing the words they say – remains a big challenge

Deep learning – building large neural networks, trained end-to-end – is proving a very powerful approach to hard artificial intelligence challenges

Neural network learning isn't voodoo and magic!
90% of it is efficient application of the chain rule

Deep Learning for Language Understanding
Step 1: Word Vectors

Christopher Manning

Stanford University

@chrmanning ❀ @stanfordnlp

Harker Programming Invitational 2017