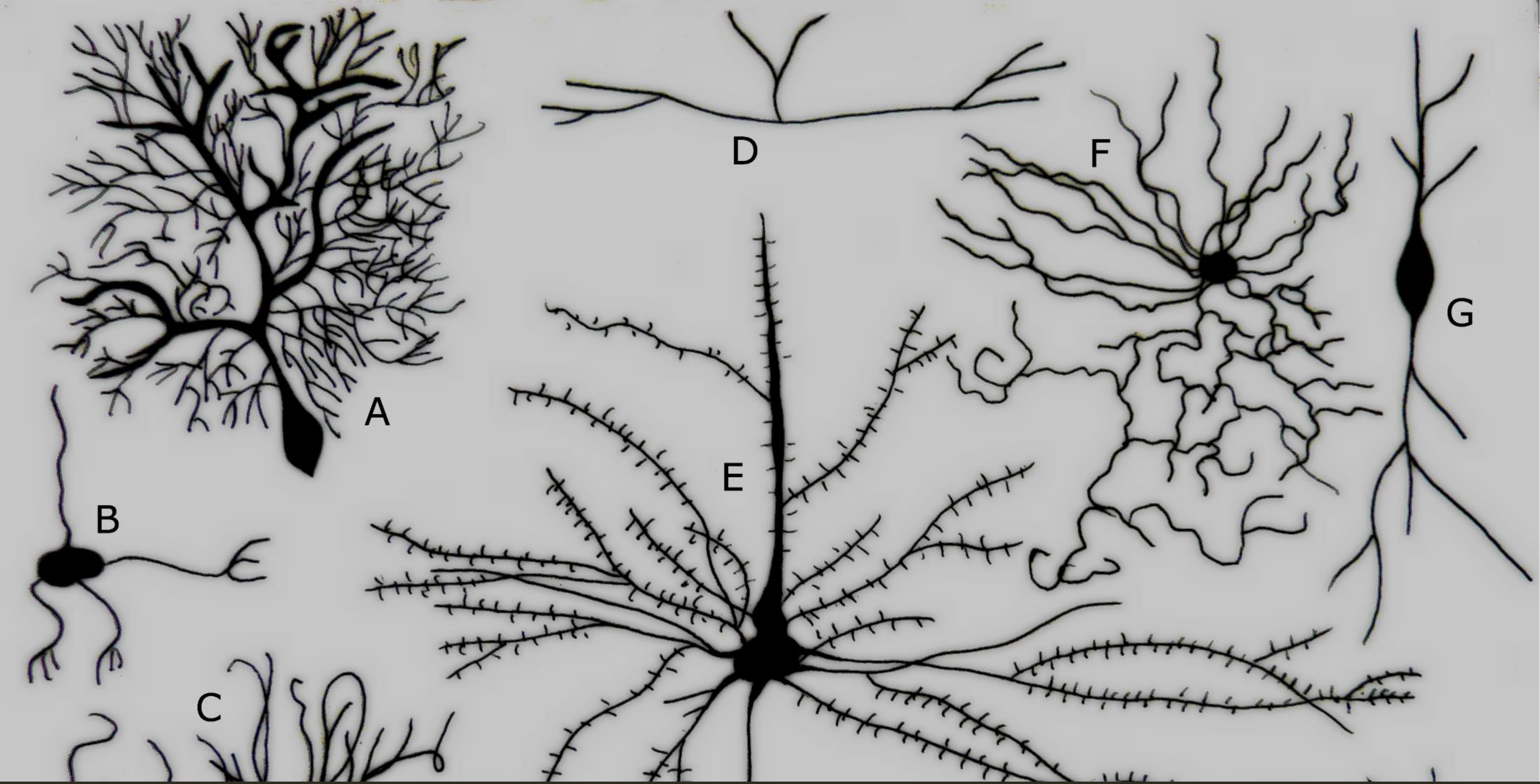


10 NEURAL MODELS OF WORD REPRESENTATION

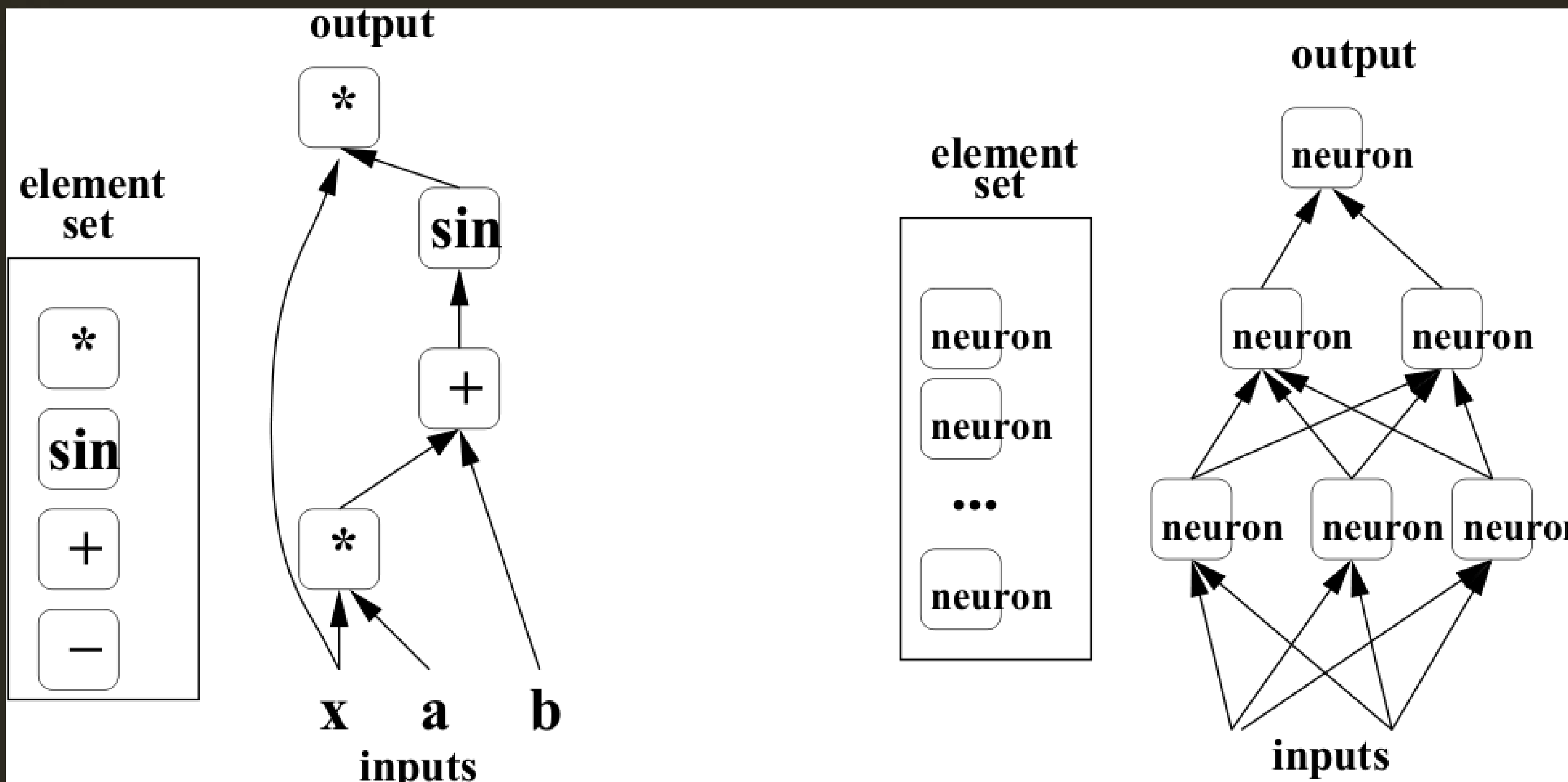
CSC2501/485 Fall 2015

Frank Rudzicz
Toronto Rehabilitation Institute;
University of Toronto

DEEP MOTIVATIONS

- Brains have a deep architecture.
- Humans organize their ideas hierarchically, through composition of simpler ideas.
- Insufficiently deep architectures can be exponentially inefficient.
- Distributed (possibly sparse) representations are necessary to achieve non-local generalization.
- Multiple levels of latent variables allow combinatorial sharing of statistical strength.

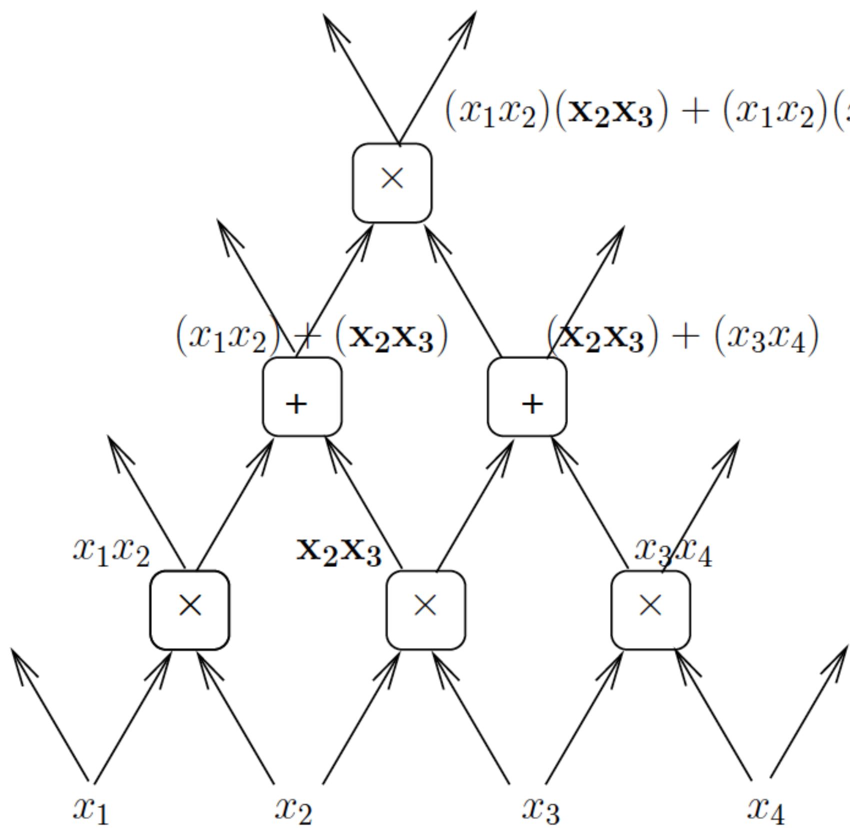
ARCHITECTURAL DEPTH



Depth = 4

Depth = 3

ARCHITECTURAL DEPTH



Polynomial expressed with shared components:

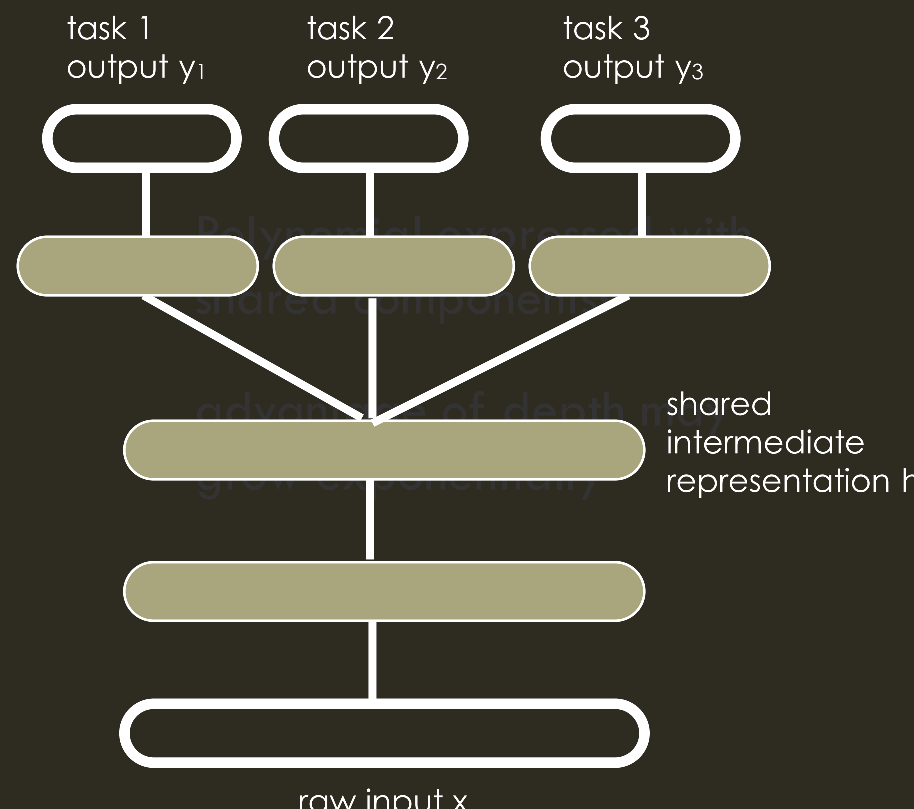
advantage of depth may grow exponentially

GENERALIZATION FROM DEPTH

Generalizing better to new tasks is crucial to AI

Deep architectures learn good *intermediate representations* that can be shared across tasks

A good representation is one that makes sense for many tasks



From Bengio (2009)

CLASSIC CL – MEANING

```
Python 3.4.1... on win32
>>> from nltk.corpus import wordnet as wn
>>> platypus = wn.synset('platypus.n.01')
>>> hyper = lambda s: s.hypernyms()
>>> list(platypus.closure(hyper))
[Synset('monotreme.n.01'), Synset('prototherian.n.01'), Synset('mammal.n.01'),
 Synset('vertebrate.n.01'), Synset('chordate.n.01'), Synset('animal.n.01'),
 Synset('organism.n.01'), Synset('living_thing.n.01'), Synset('whole.n.02'),
 Synset('object.n.01'), Synset('physical_entity.n.01'), Synset('entity.n.01')]
>>>
```

Well, this sort of representation *can* be applied to many different tasks…

CLASSIC CL – LEARNING

Classic NLP

Input → Feature representation → Machine learning → Output

Task: find all verbs in a sentence

- Manually define a good, meaningful representation
  - E.g., ends on -ed, -ing, +front/high vowel
- But what about spelling mistakes? Or slang?
  - E.g., ends on -edd, -in, -inn,…
- You can NEVER define all features manually!

He is $\boxed{\text{walking}}$ away…

↓

Simple tokenization

↓

Morphological analysis

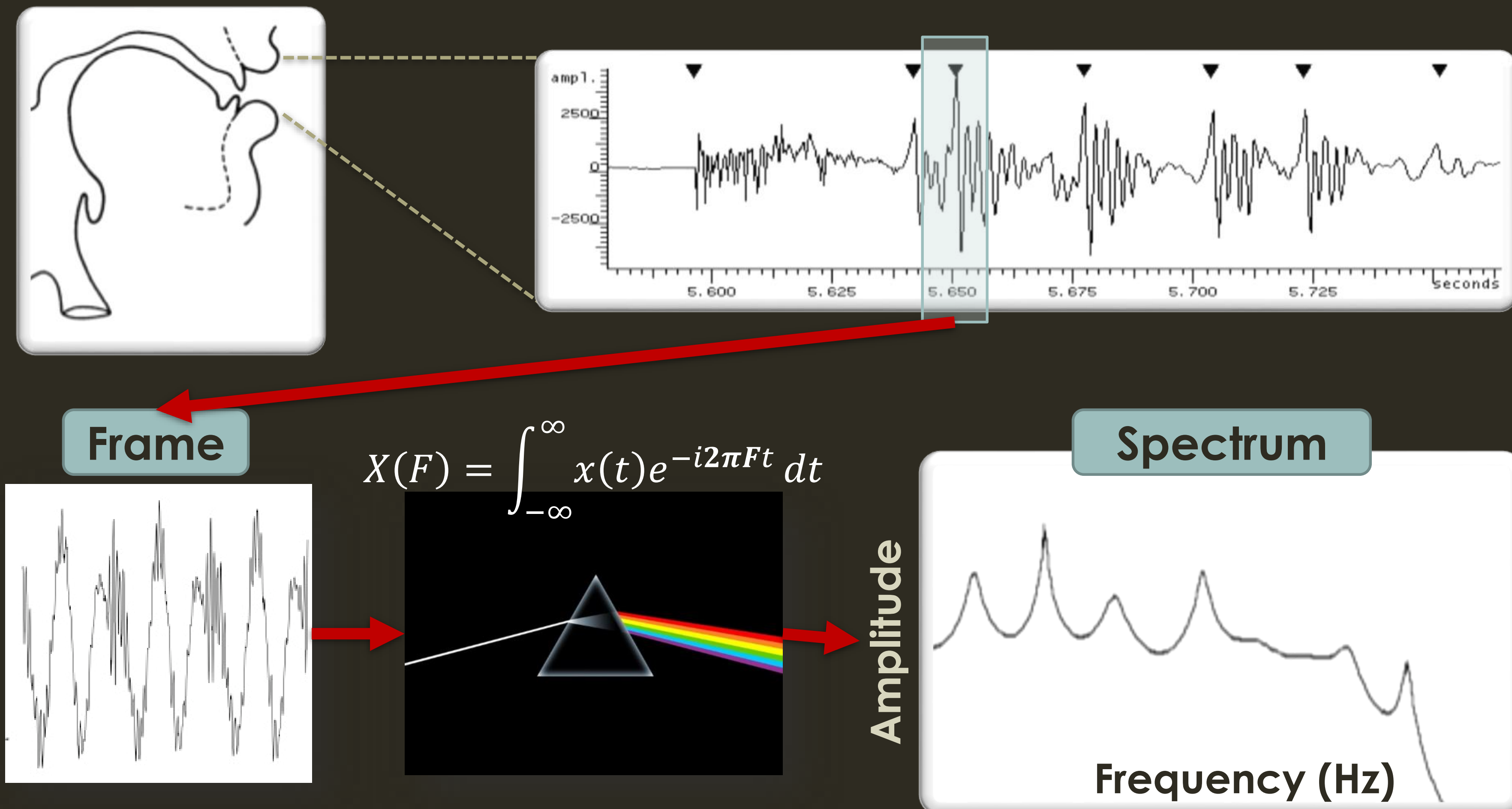
↓

Syntactic analysis

↓

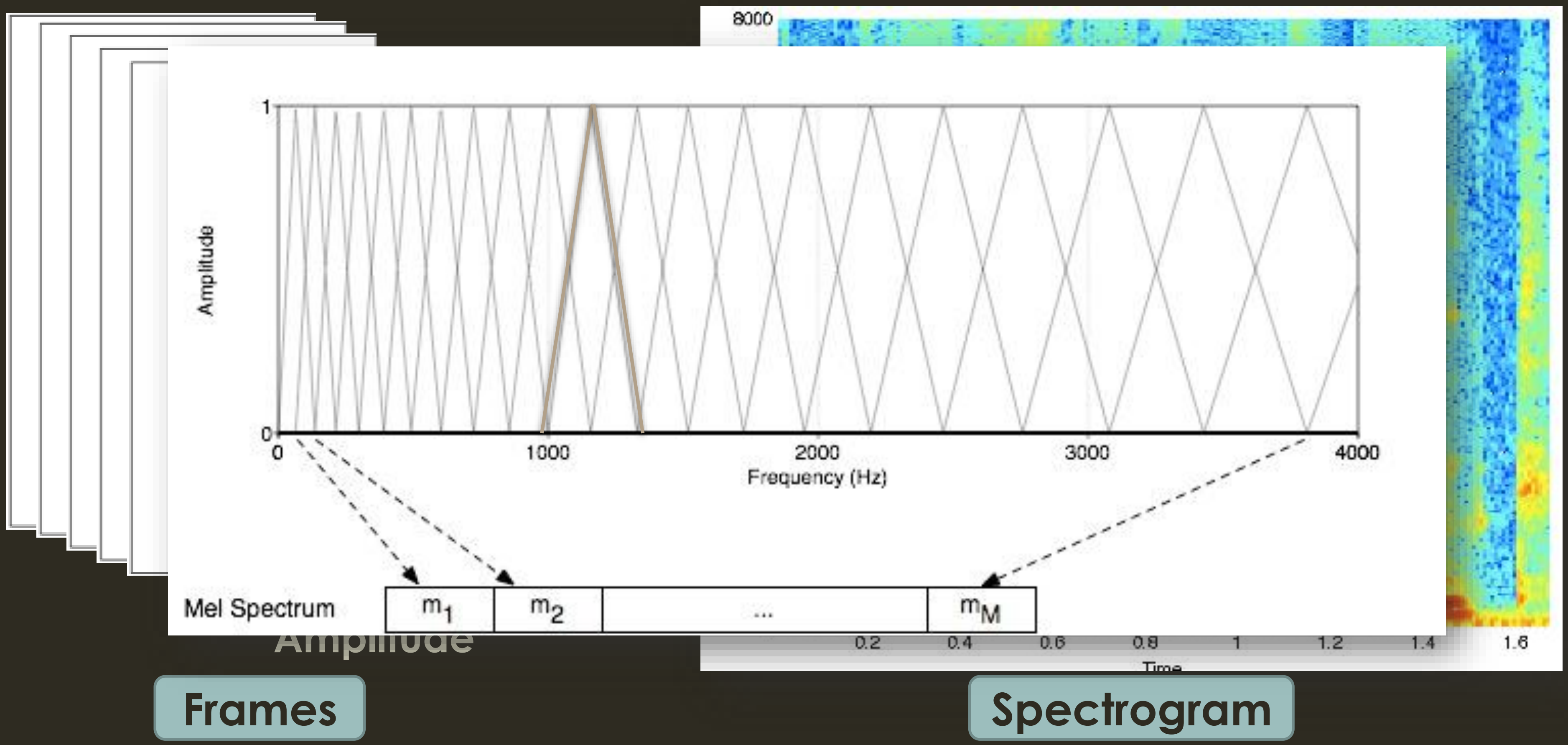
Verb, present particle

CLASSIC SPEECH

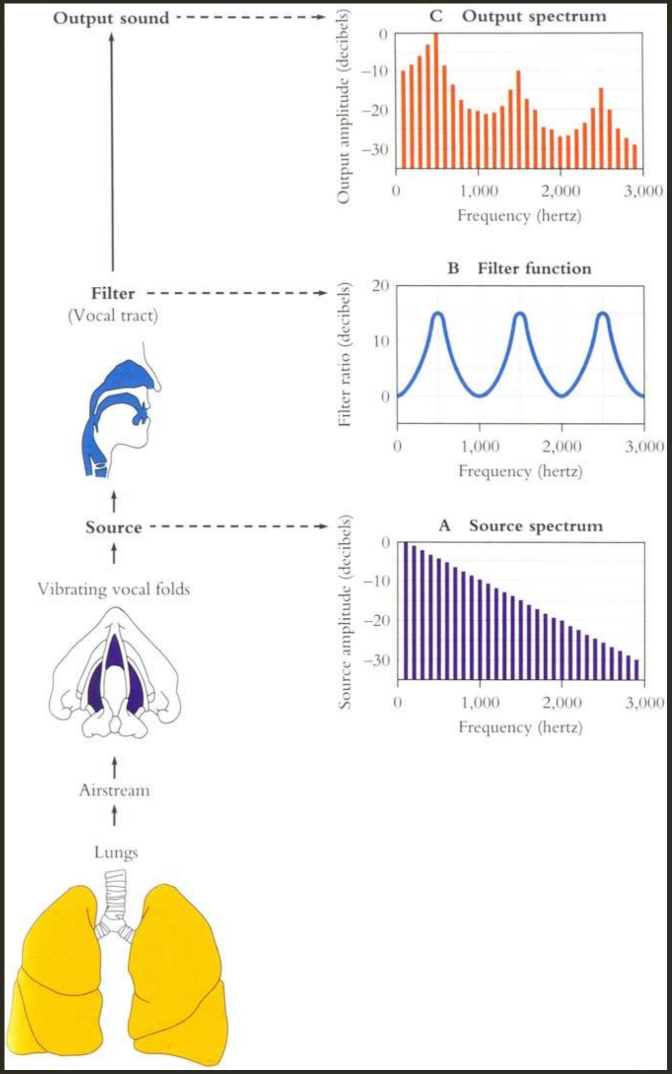


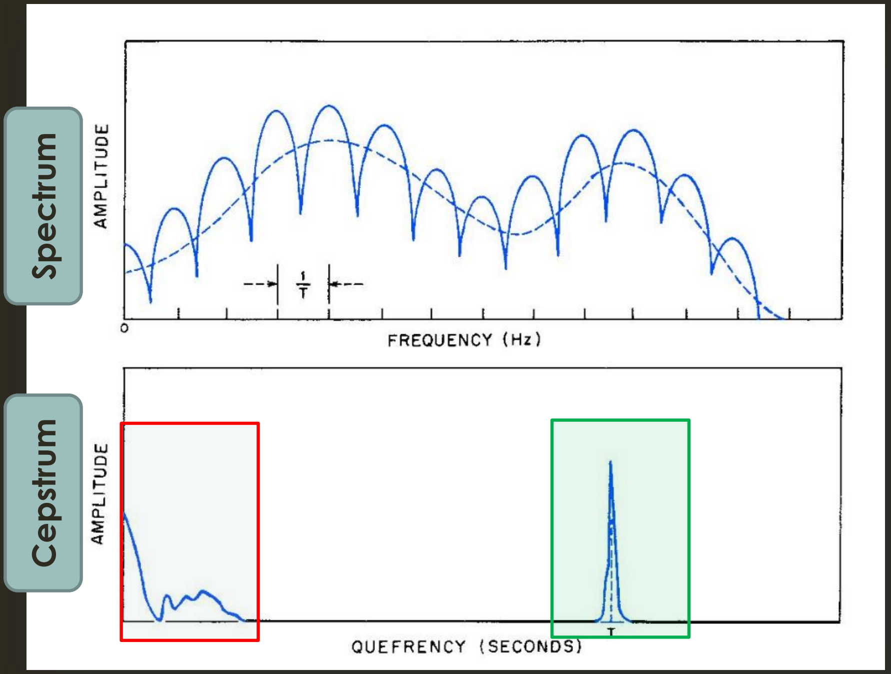
$$X(F) = \int_{-\infty}^{\infty} x(t) e^{-i2\pi F t}\, dt$$

CLASSIC SPEECH



CLASSIC SPEECH





This is due to the vocal tract shape

This is due to the glottis

Pictures from John Coleman (2005)

NEW CL

Classic NLP

Input → Feature representation → Machine learning → Output

Deep learning NLP

Input → Feature representation → Machine learning → Output

Automatically learn the feature representation, too!
(because it's 2015)

He is walking away…

↓

Simple tokenization

↓

Morphological analysis

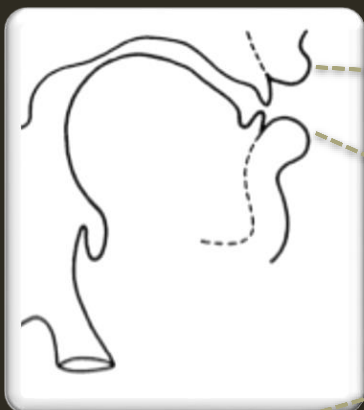
↓

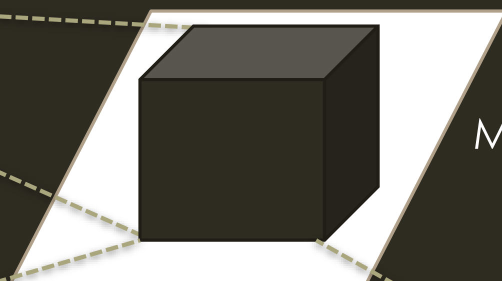
Syntactic analysis

↓

Verb, present particle

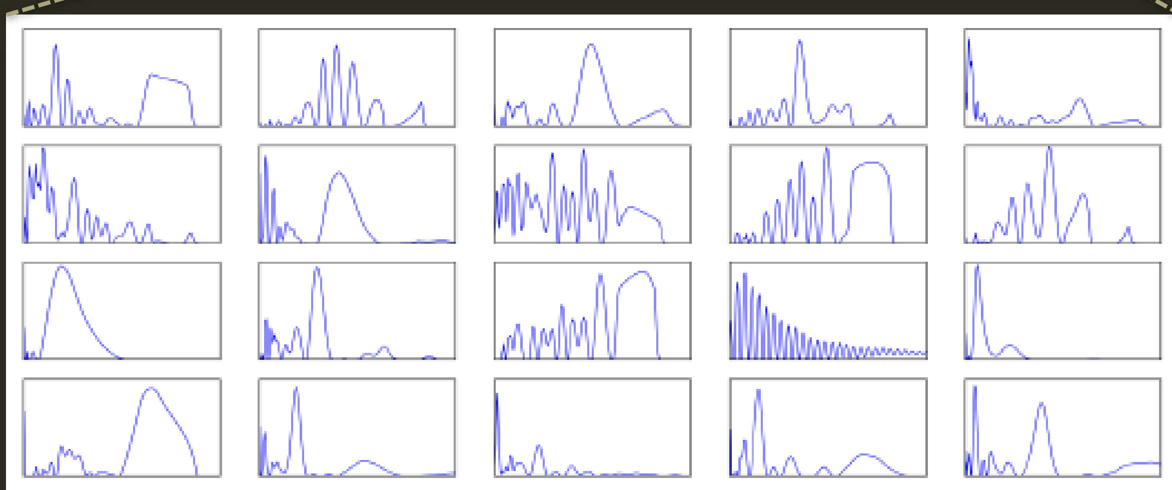
DEEP LEARNING IN SPEECH





Magic deep thingie

"We have no idea how speech works"
- [someone from Stanford]



From Jaitly (2014)

Feat.	Type	RT03S FSH	Hub5 SWB
Trad.	1-pass adapt	27.4%	23.6%
Deep	1-pass adapt	18.5% (-33%)	16.1% (-32%)

From Socher (2015)

WORDS

- Given a corpus with D (e.g., $= 100K$) unique words, the **classical binary approach** is to uniquely assign **each word** with an index in D-dimensional vectors ('one-hot' representation).

0	0	0	0	..	0	1	0	…	0

D

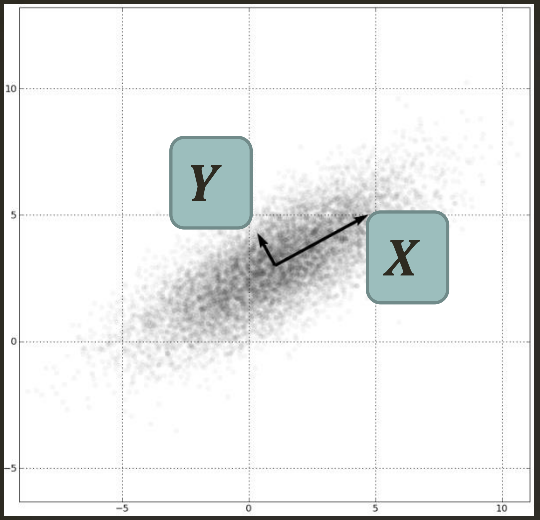
- Classic word-feature representation assigns **features** to each index.
  - E.g., 'VBG', 'positive', 'age-of-acquisition'.

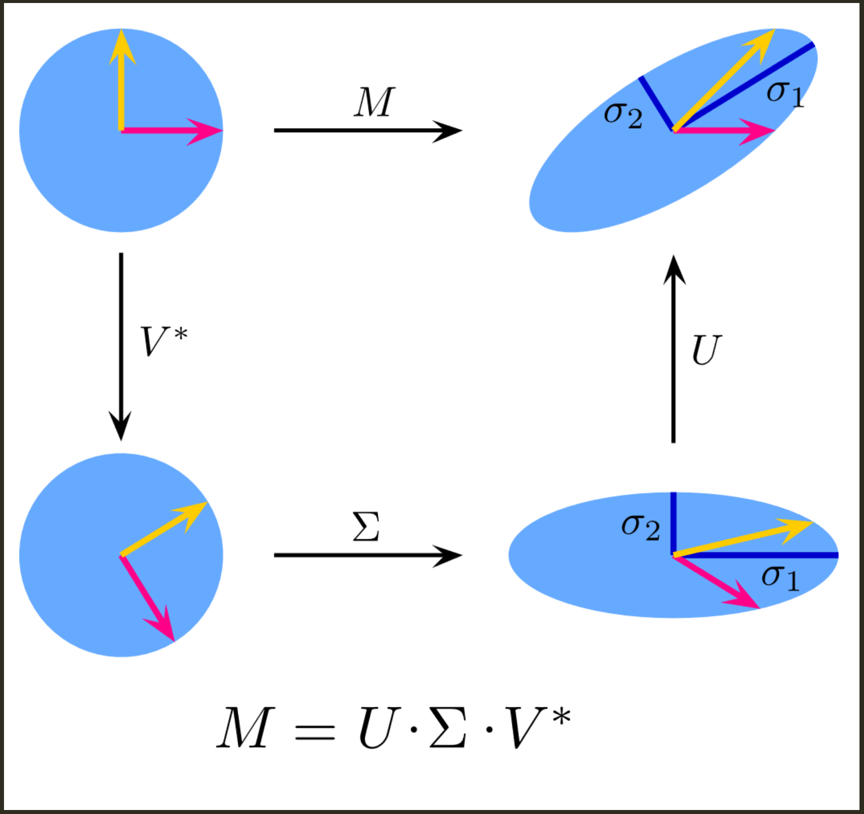
1	0.8	4.5	0.81	…	99

$d \ll D$

- Is there a way to learn something *like* the latter?

SINGULAR VALUE DECOMPOSITION





PCA

SVD

SINGULAR VALUE DECOMPOSITION

Corpus

	a	as	chuck	could	how	if	much	wood	woodch.	would	,	.	?	a	as	chuck	could	how	if	much	wood	woodch.	would	,	.	?
a	13	24	12	3	9	20	22	31	16	23	18	0	7	13	7	31	26	0	14	4	21	50	9	16	7	7
as	7	8	15	11	0	5	9	25	10	0	3	0	17	24	8	2	3	0	9	10	10	20	13	11	0	0
chuck	31	2	5	20	5	14	6	9	36	15	12	0	0	12	15	5	6	0	9	8	30	10	2	11	9	12
could	26	3	6	0	0	16	2	4	30	9	14	0	0	3	11	20	0	0	0	6	23	2	1	0	8	8
how	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	5	0	0	3	10	9	7	8	4	0	0
if	14	9	9	0	3	0	8	11	16	15	20	0	2	20	5	14	16	0	0	3	14	18	0	0	5	5
much	4	10	8	6	10	3	0	8	5	0	2	0	9	22	9	6	2	0	8	0	20	18	15	10	0	0
wood	21	10	30	23	9	14	20	7	26	5	11	0	8	31	25	9	4	0	11	8	7	26	20	14	10	10
woodch.	50	20	10	2	7	18	18	26	13	20	16	0	5	16	10	36	30	0	16	5	26	13	10	18	9	9
would	9	13	2	1	8	0	15	20	10	0	0	0	4	23	0	15	9	0	15	0	5	20	0	17	3	0
,	16	11	11	0	4	0	10	14	18	17	0	0	3	18	3	12	14	0	20	2	11	16	0	0	4	4
.	7	0	9	8	0	5	0	10	9	3	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
?	7	0	12	8	0	5	0	10	9	0	4	0	0	7	17	0	0	0	2	9	8	5	4	3	0	0

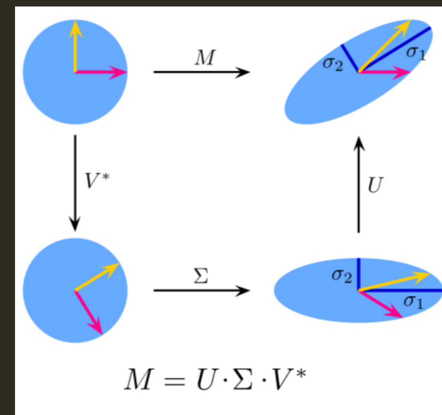
Co-occurrence

Rohde *et al.* (2006) An Improved Model of Semantic Similarity Based on Lexical Co-Occurrence. *Communications of the ACM* **8**:627-633.

SINGULAR VALUE DECOMPOSITION

$M =$

	a	as	chuck	could	how	if	much	wood	woodch.	would	,	.	?	a	as	chuck	could	how	if	much	wood	woodch.	would	,	.	?
a	13	24	12	3	9	20	22	31	16	23	18	0	7	13	7	31	26	0	14	4	21	50	9	16	7	7
as	7	8	15	11	0	5	9	25	10	0	3	0	17	24	8	2	3	0	9	10	10	20	13	11	0	0
chuck	31	2	5	20	5	14	6	9	36	15	12	0	0	12	15	5	6	0	9	8	30	10	2	11	9	12
could	26	3	6	0	0	16	2	4	30	9	14	0	0	3	11	20	0	0	0	6	23	2	1	0	8	8
how	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	5	0	0	3	10	9	7	8	4	0	0
if	14	9	9	0	3	0	8	11	16	15	20	0	2	20	5	14	16	0	0	3	14	18	0	0	5	5
much	4	10	8	6	10	3	0	8	5	0	2	0	9	22	9	6	2	0	8	0	20	18	15	10	0	0
wood	21	10	30	23	9	14	20	7	26	5	11	0	8	31	25	9	4	0	11	8	7	26	20	14	10	10
woodch.	50	20	10	2	7	18	18	26	13	20	16	0	5	16	10	36	30	0	16	5	26	13	10	18	9	9
would	9	13	2	1	8	0	15	20	10	0	0	0	4	23	0	15	9	0	15	0	5	20	0	17	3	0
,	16	11	11	0	4	0	10	14	18	17	0	0	3	18	3	12	14	0	20	2	11	16	0	0	4	4
.	7	0	9	8	0	5	0	10	9	3	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
?	7	0	12	8	0	5	0	10	9	0	4	0	0	7	17	0	0	0	2	9	8	5	4	3	0	0



$$A = U_{[:,1:2]}\Sigma_{[1:2,1:2]}$$

$U =$

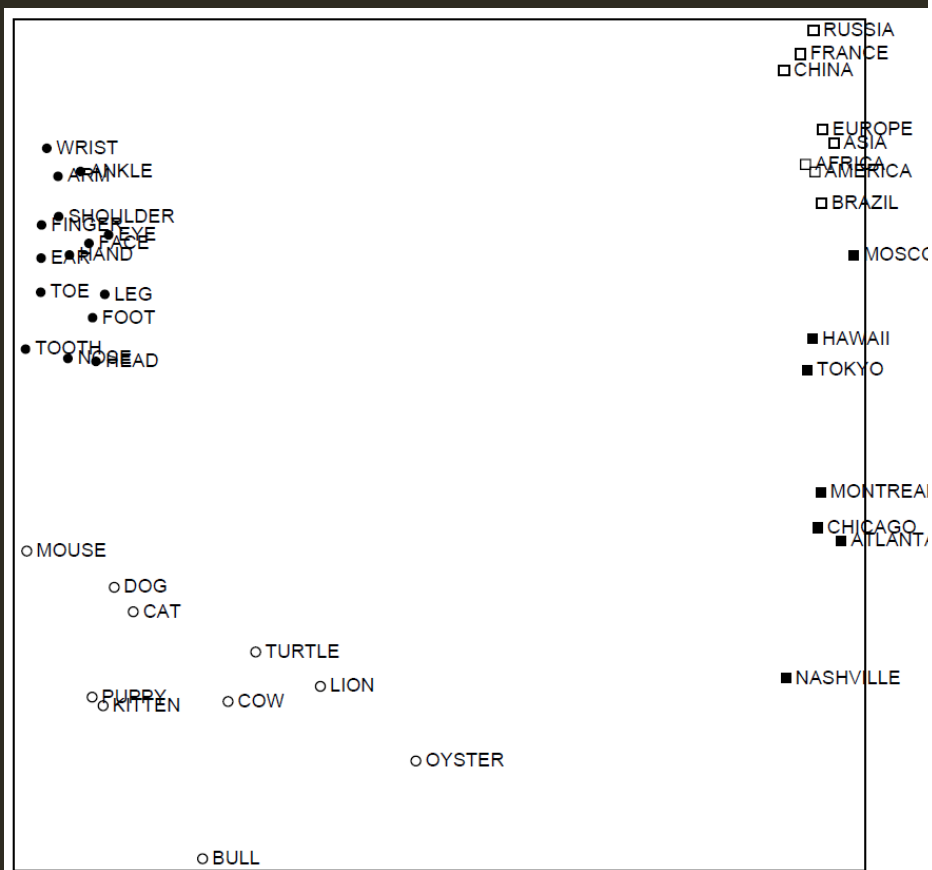
a	-0.44	-0.30	0.57	0.58	…
as	-0.13	-0.33	-0.59	0	…
chuck	-0.48	-0.51	-0.37	0	…
could	-0.70	0.35	0.15	-0.58	…
…	…	…	…	…	…

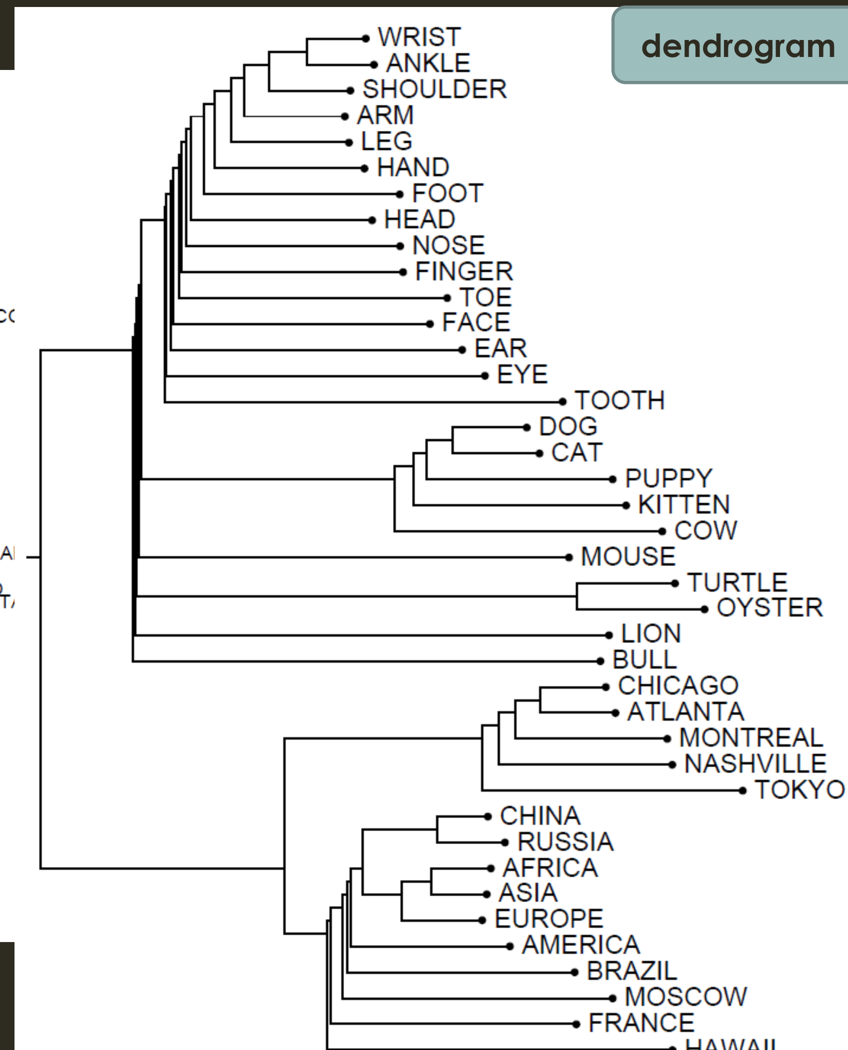
$\Sigma =$

2.16	0	0	0	…
0	1.59	0	0	…
0	0	1.28	0	…
0	0	0	1	…
…	…	…	…	…

Rohde *et al.* (2006) An Improved Model of Semantic Similarity Based on Lexical Co-Occurrence. *Communications of the ACM* **8**:627-633.

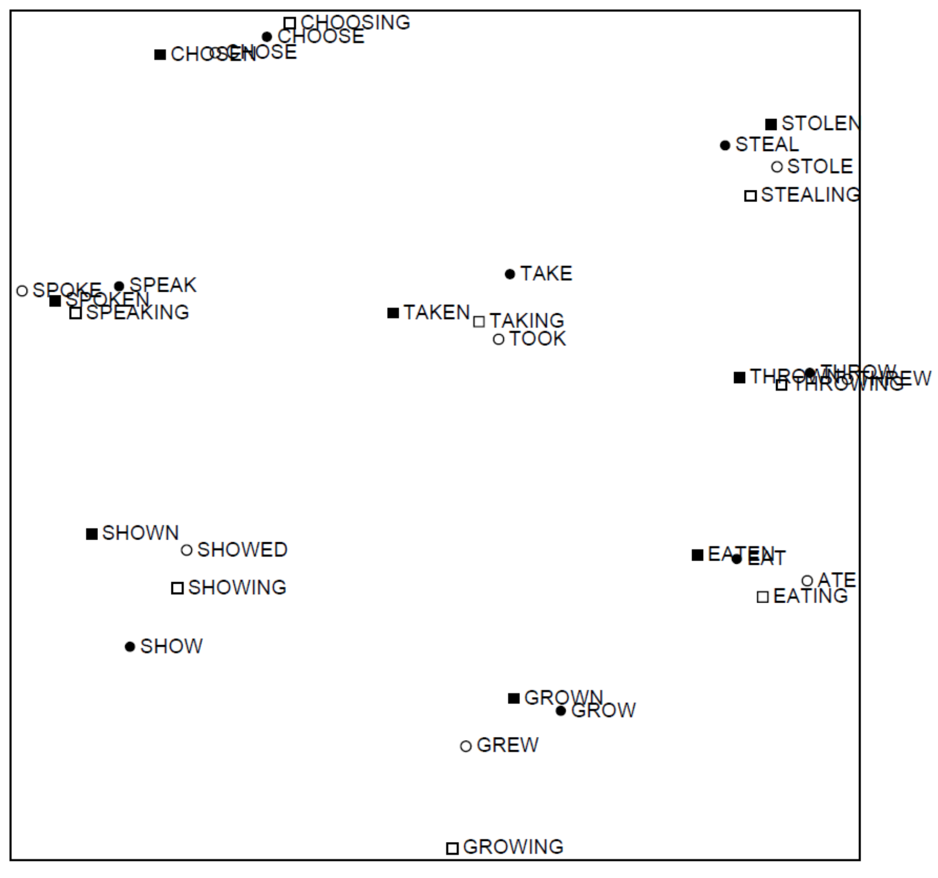
SINGULAR VALUE DECOMPOSITION





Rohde *et al.* (2006) An Improved Model of Semantic Similarity Based on Lexical Co-Occurrence. *Communications of the ACM* **8**:627-633.

SINGULAR VALUE DECOMPOSITION



Rohde *et al*. (2006) An Improved Model of Semantic Similarity Based on Lexical Co-Occurrence. *Communications of the ACM* **8**:627-633.

PROBLEMS WITH SVD; INTRO TO WORD2VEC

- SVD: Computational costs scale quadratically with M.
  'Hard' to incorporate new words.

- Word2vec: Don't capture co-occurrence directly
  Just try to predict surrounding words, baby.

$$P(w_{t+1} = yourself \mid w_t = kiss)$$

you	go	kiss	yourself	,
you	go	hug	yourself	,
		…		

'softmax'

$$P(w_o|w_i) = \frac{\exp(V_{w_o}^{\intercal} v_{w_i})}{\sum_{w=1}^{W} \exp(V_w^{\intercal} v_{w_i})}$$

Where v_w is the 'input' vector for word w,
and V_w is the 'output' vector for word w,

https://code.google.com/p/word2vec/

LEARNING WORD REPRESENTATIONS

- Word representations can be learned using the following **objective function:**

$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T} \sum_{-c<j<c, j\neq 0} \log P(w_{t+j}|w_t)$$

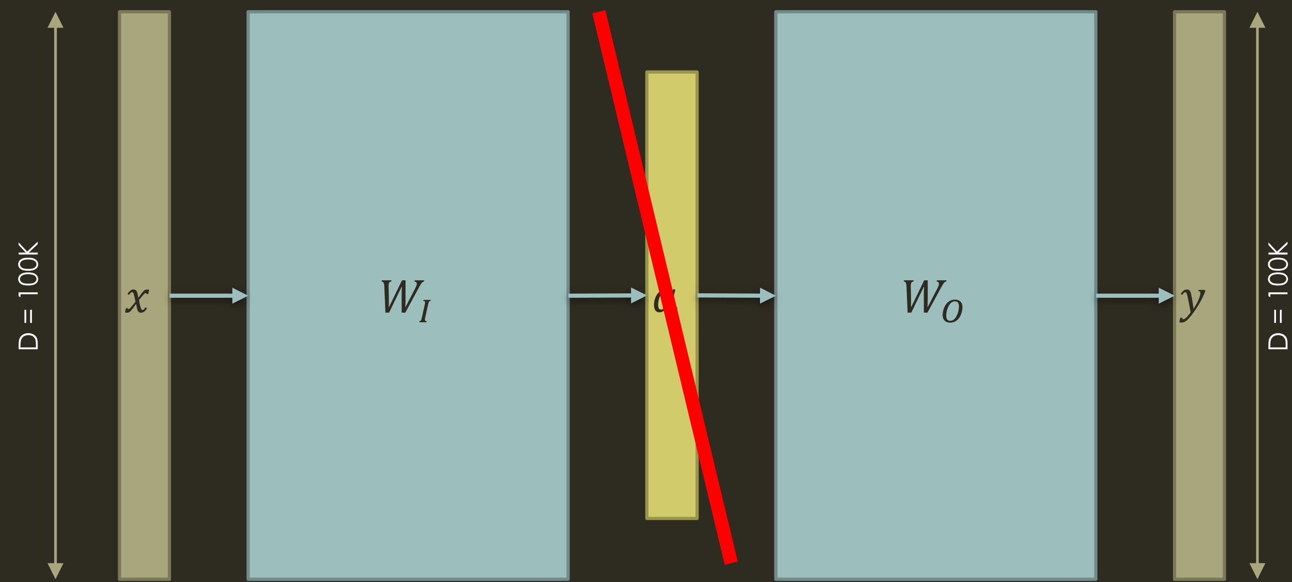
where w_t is the t^{th} word in a sequence of T words.

- This is closely related to **word prediction.**
  - *"words of a feather flock together."*
  - *"you shall know a word by the company it keeps."*

    - J.R. Firth (1957)

go	kiss	yourself
go	hug	yourself
	…	

LEARNING WORD REPRESENTATIONS



$[0,0,0, \dots 1, \dots, 0]$
kiss

"outside"	"inside"	"outside"
go	kiss	yourself
go	hug	yourself
	…	

$[0,1,0, \dots, 0, \dots, 0]$ go
$[0,0,1, \dots, 0, \dots, 0]$ yourself

Note: we now have two representations of each word:
v_w comes from the rows of W_I
V_w comes from the cols of W_O

Continuous bag of words (CBOW)

USING WORD REPRESENTATIONS

Without a latent space,

kiss = $[0,0,0, \dots, 0,1,0, \dots, 0]$, &

hug = $[0,0,0, \dots, 0,0,1, \dots, 0]$ so

Similarity = $\cos(x, y) = 0.0$

Transform

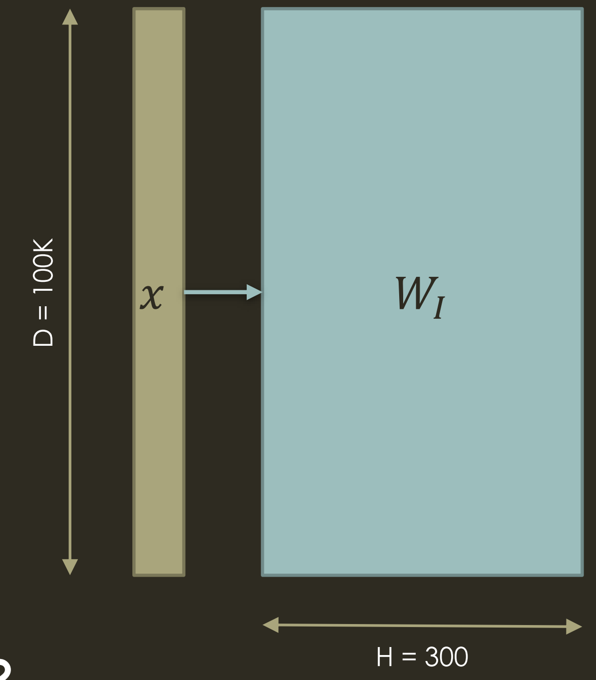
$v_w = xW_1$

In latent space,

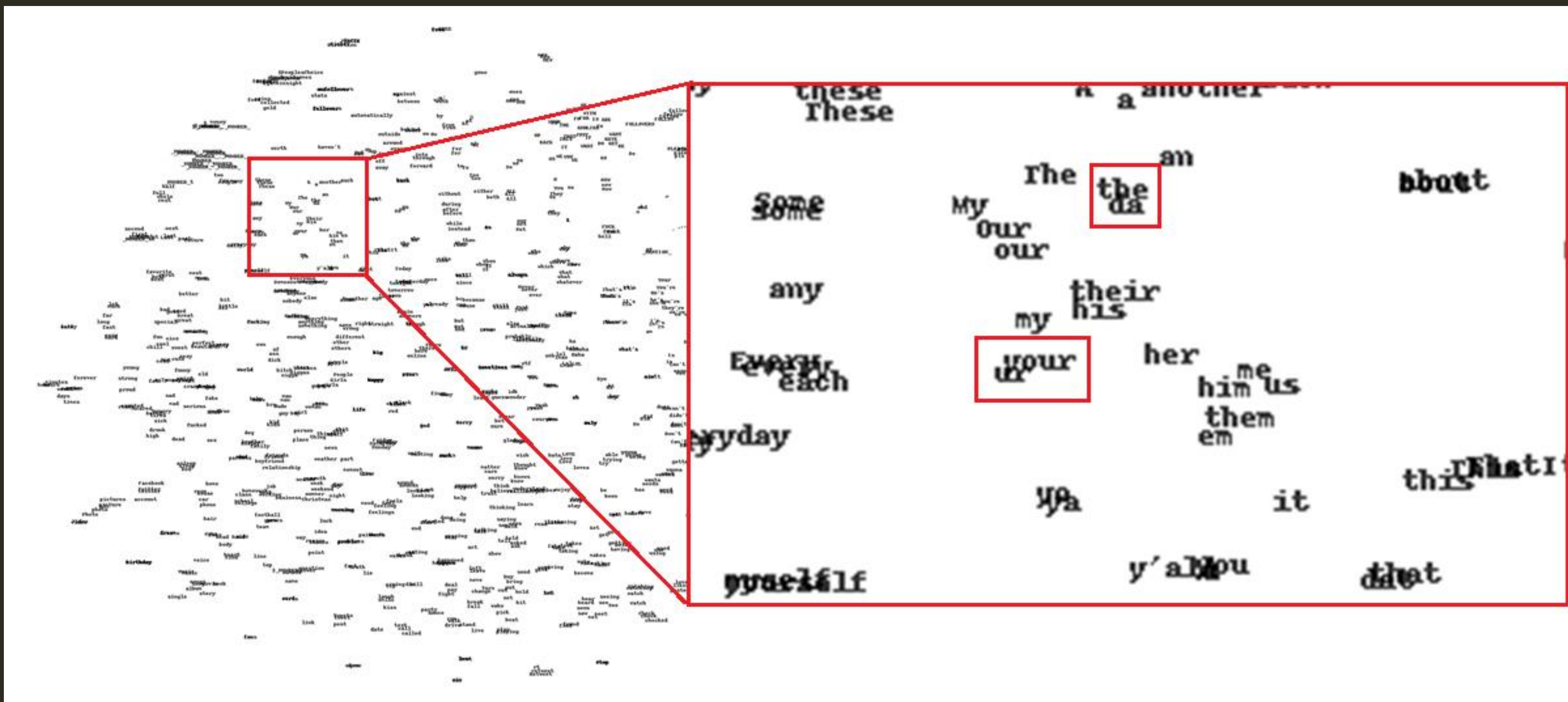
kiss = $[0.8,0.69,0.4, \dots, 0.05]_H$, &

hug = $[0.9,0.7,0.43, \dots, 0.05]_H$ so

Similarity = $\cos(x, y) = 0.9$



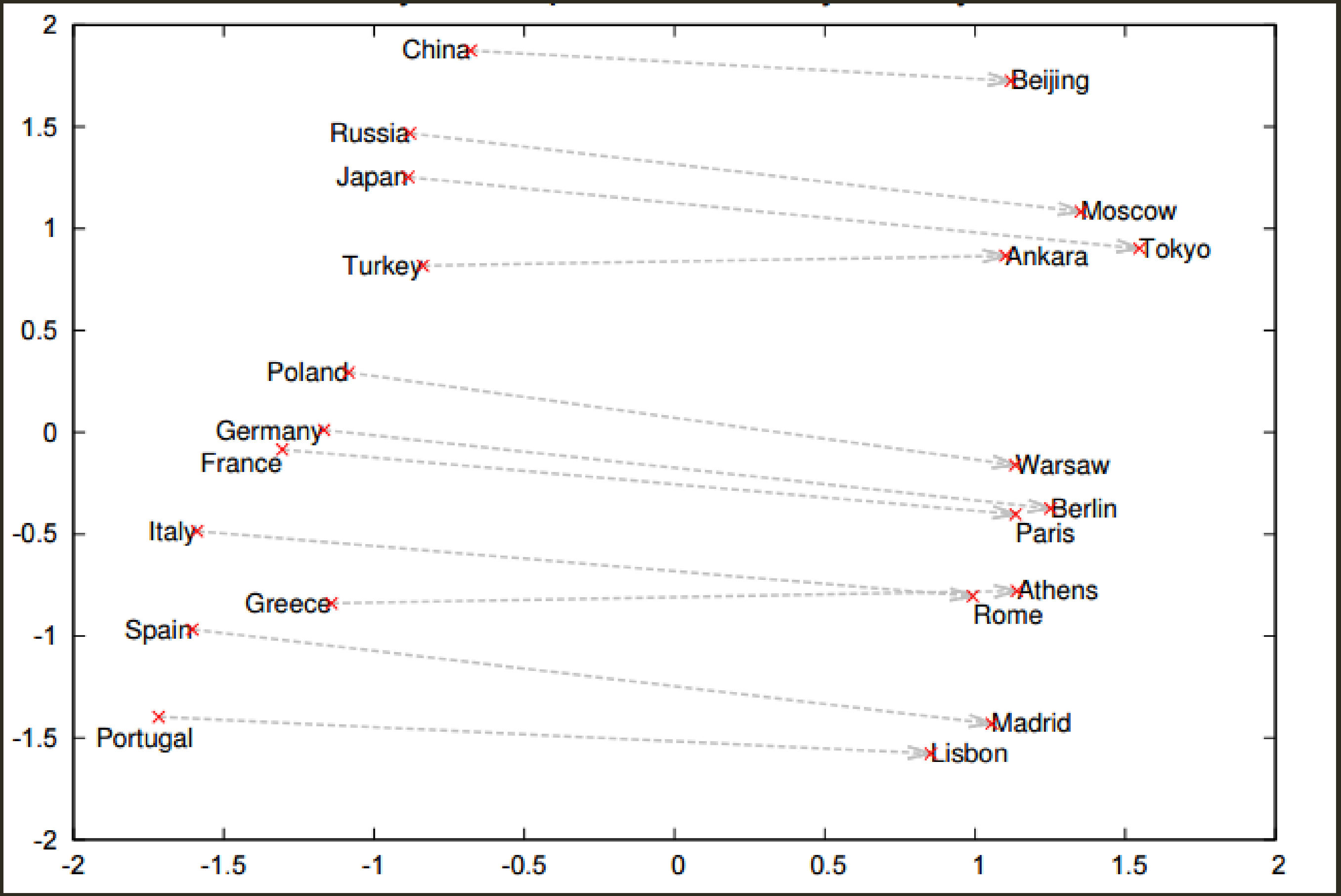
LINGUISTIC REGULARITIES IN WORD-VECTOR SPACE



Visualization of a vector space of the top 1000 words in Twitter

Trained on 400 million tweets having 5 billion words

LINGUISTIC REGULARITIES IN WORD-VECTOR SPACE



Trained on the Google news corpus with over 300 billion words.

LINGUISTIC REGULARITIES IN WORD-VECTOR SPACE

Expression	Nearest token
Paris – France + Italy	Rome
Bigger – big + cold	Colder
Sushi – Japan + Germany	bratwurst
Cu – copper + gold	Au
Windows – Microsoft + Google	Android

Analogies: apple:apples :: octopus:octopodes
Hypernymy: shirt:clothing :: chair:furniture

Ha ha – isn't that nice? But it's easy to cherry-pick…

ACTUALLY DOING THE LEARNING

First, let's define what our parameters are.
Given H-dimensional vectors, and V words:

$$\theta = \begin{bmatrix} v_a \\ v_{aardvark} \\ \vdots \\ v_{zymurgy} \\ V_a \\ V_{aardvark} \\ \vdots \\ V_{zymurgy} \end{bmatrix} \in \mathbb{R}^{2VH}$$

ACTUALLY DOING THE LEARNING

Many options. Gradient descent is popular.

We want to optimize

$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T} \sum_{-c<j<c, j\neq 0} \log P(\overset{\text{“outside”}}{w_{t+j}} | \overset{\text{“inside”}}{w_t})$$

And we want to update vectors $V_{w_{t+j}}$ then v_{w_t} within θ

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

so we'll need to take the derivative of the (log of the) softmax function:

$$P(w_{t+j}|w_t) = \frac{\exp(V_{w_{t+j}}^\intercal v_{w_t})}{\sum_{w=1}^{W} \exp(V_w^\intercal v_{w_t})}$$

ACTUALLY DOING THE LEARNING

We need to take the derivative of the (log of the) softmax function:

$$\frac{\delta}{\delta v_{w_t}} \log P\left(w_{t+j} \middle| w_t\right) = \frac{\delta}{\delta v_{w_t}} \log \frac{\exp(V_{w_{t+j}}^{\intercal} v_{w_t})}{\sum_{w=1}^{W} \exp(V_w^{\intercal} v_{w_t})}$$

$$= \frac{\delta}{\delta v_{w_t}} \log \exp\left(V_{w_{t+j}}^{\intercal} v_{w_t}\right) - \log \sum_{w=1}^{W} \exp(V_w^{\intercal} v_{w_t})$$

$$= V_{w_{t+j}} - \frac{\delta}{\delta v_{w_t}} \log \sum_{w=1}^{W} \exp(V_w^{\intercal} v_{w_t})$$

[apply the chain rule $\frac{\delta f}{\delta v_{w_t}} = \frac{\delta f}{\delta z} \frac{\delta z}{\delta v_{w_t}}$]

$$= V_{w_{t+j}} - \sum_{w=1}^{W} p(w|w_t) V_w$$

More details: http://arxiv.org/pdf/1411.2738.pdf

SMELL THE GLOVE

Global Ve ctors for Word representations is a popular alternative to word2vec.

Trained on the non-zero entries of a global word-word co-occurrence matrix.

$$J(\theta) = \frac{1}{2}\sum_{ij} f\left(P_{ij}\right)\left(w_i \cdot \widetilde{w_j} - \log P_{ij}\right)^2$$

Fast and scalable.

Same kinds of benefits

Words close to *frog*

3. litoria

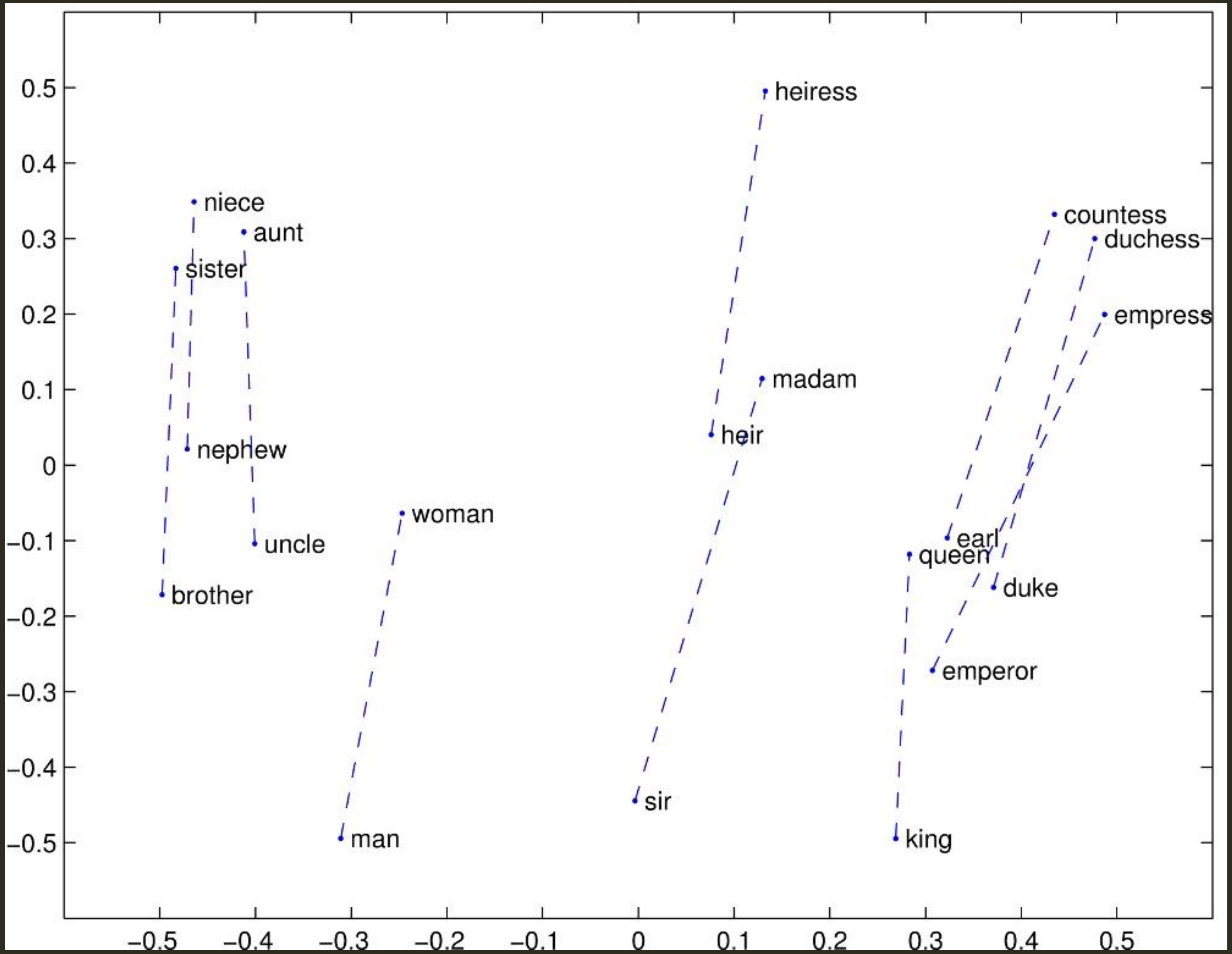
4. leptodactylidae

5. rana

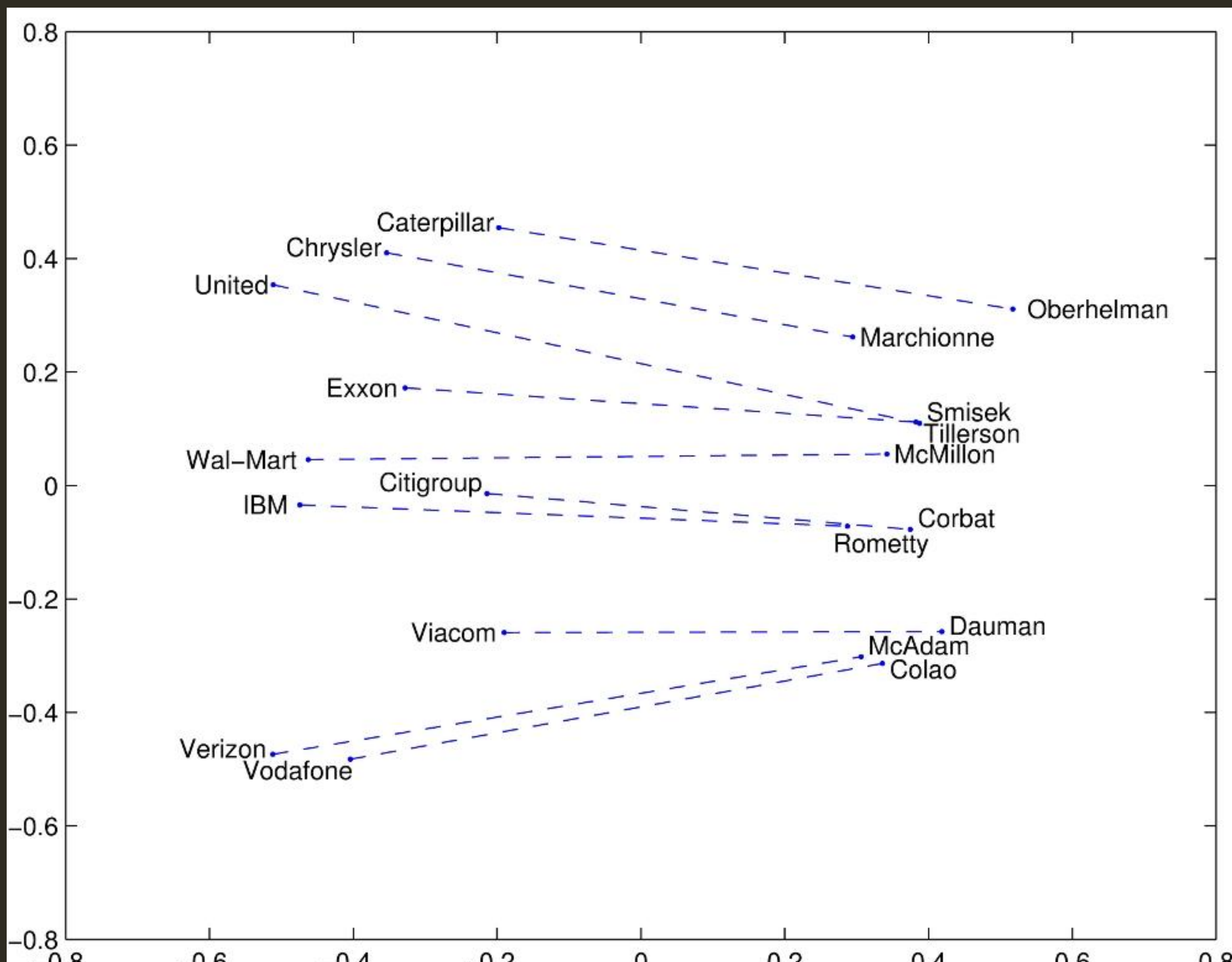
7. eleutherodactylus

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

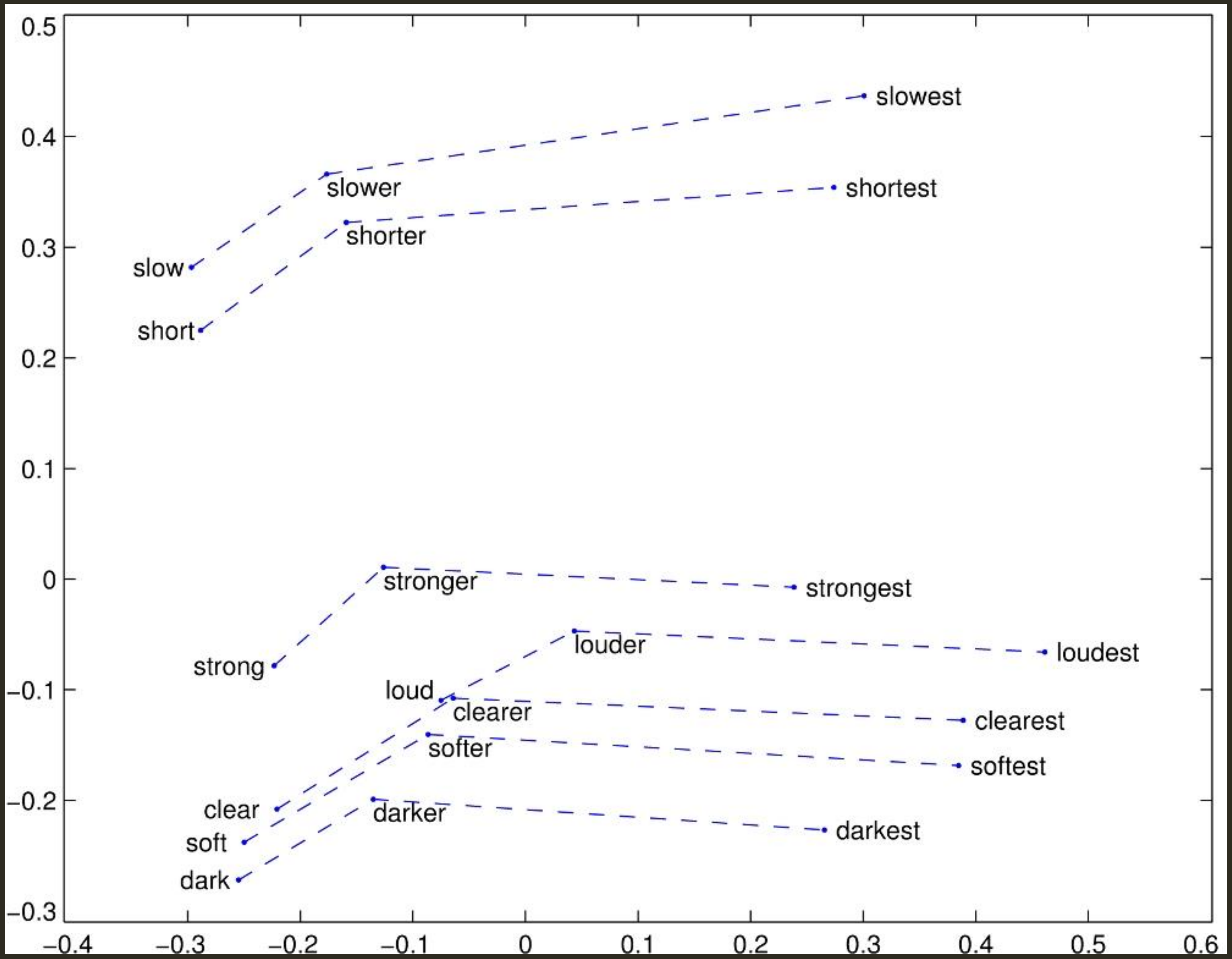
LOOK AT THE GLOVE



LOOK AT THE GLOVE



LOOK AT THE GLOVE



RESULTS – NOTE THEY'RE ALL EXTRINSIC

Bengio *et al* 2001, 2003: beating N-grams on small datasets (Brown & APNews), but much slower.

Schwenk *et al* 2002,2004,2006: beating state-of-the-art large-vocabulary speech recognizer using deep & distributed NLP model, with real-time speech recognition.

Morin & Bengio 2005, Blitzer *et al* 2005, Mnih & Hinton 2007,2009: better & faster models through hierarchical representations.

Collobert & Weston 2008: reaching or beating state-of-the-art in multiple NLP tasks (**SRL**, POS, NER, chunking) thanks to unsupervised pre-training and multi-task learning.

Bai et al 2009: ranking & semantic indexing (info retrieval).

SENTIMENT ANALYSIS

Traditional bag-of-words approach used dictionaries of happy and sad words, simple counts, and regression or simple binary classification.

But consider these:

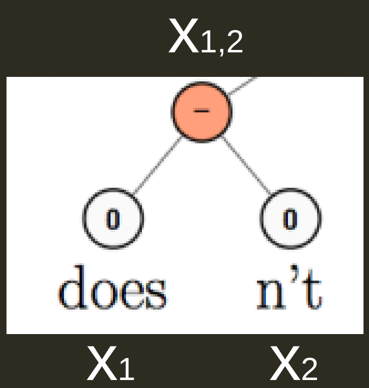
Best movie of the year

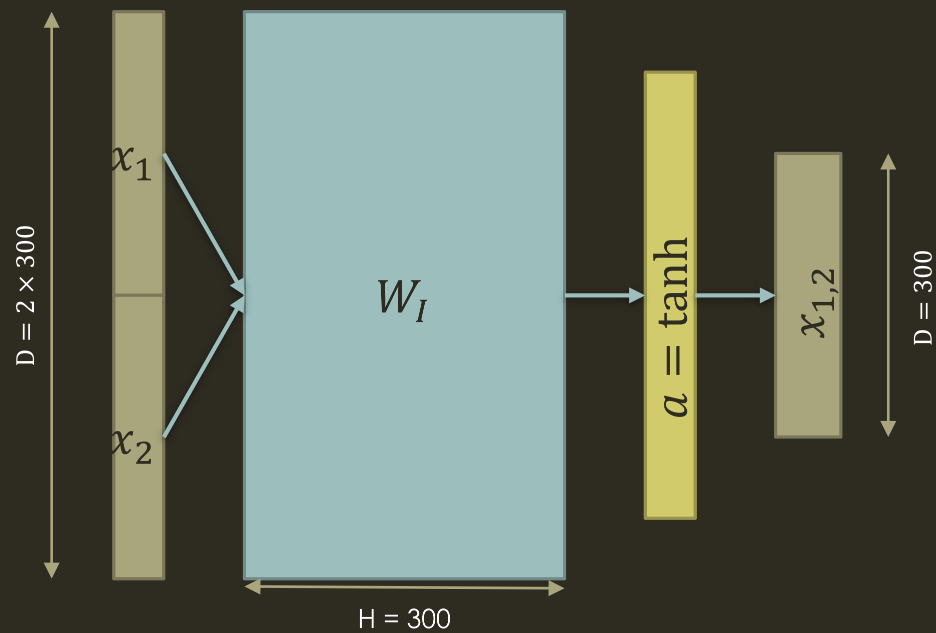
Slick and entertaining, despite a weak script

Fun and sweet but ultimately unsatisfying

SENTIMENT ANALYSIS

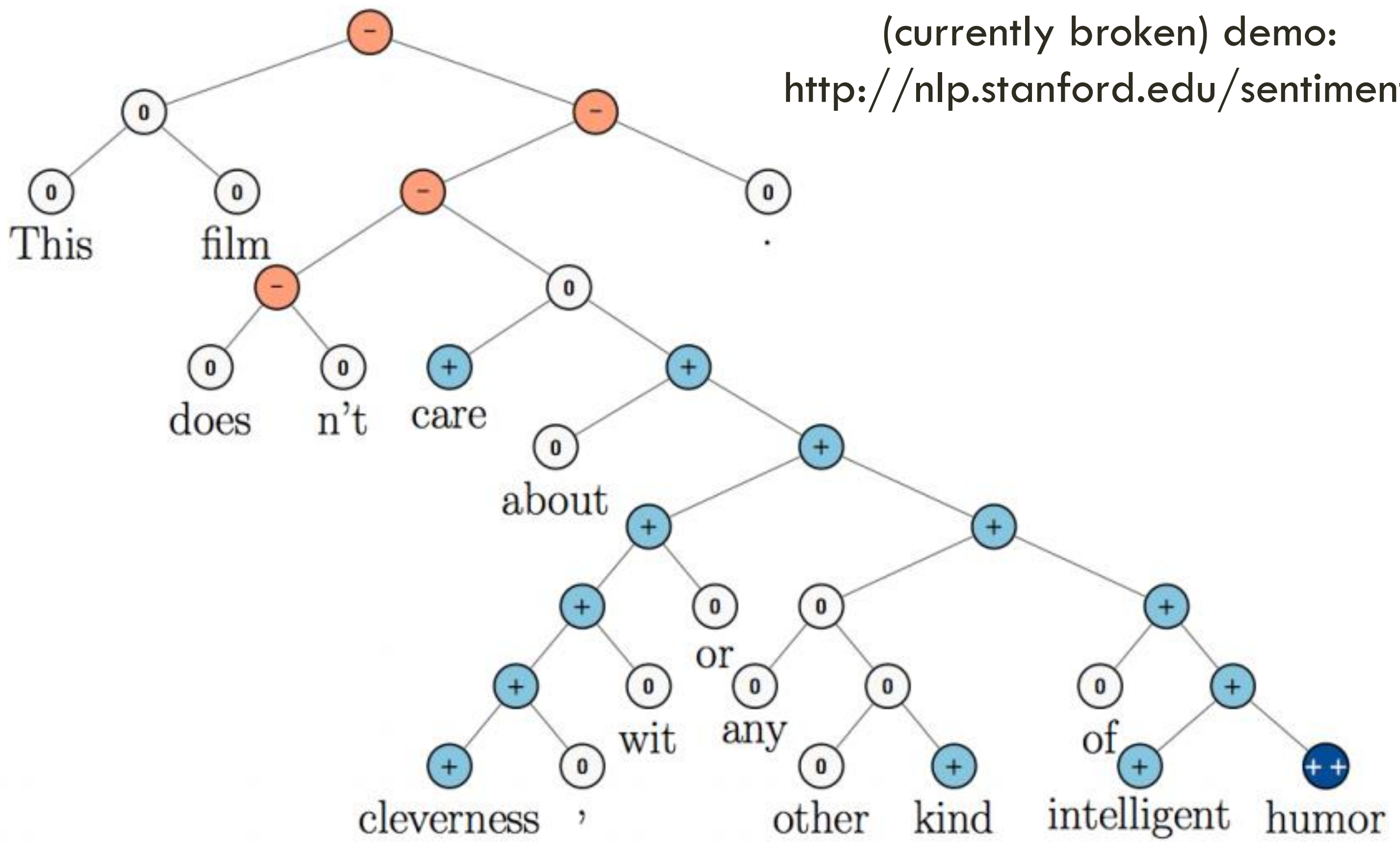
We can combine **pairs** of words into **phrase** structures. Similarly, we can combine phrase and word structures hierarchically for classification.





TREE-BASED SENTIMENT ANALYSIS

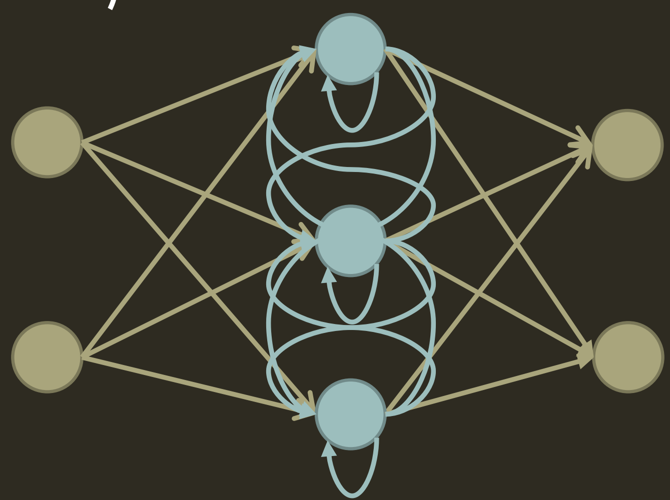
(currently broken) demo:
http://nlp.stanford.edu/sentiment/



RECURRENT NEURAL NETWORKS (RNNS)

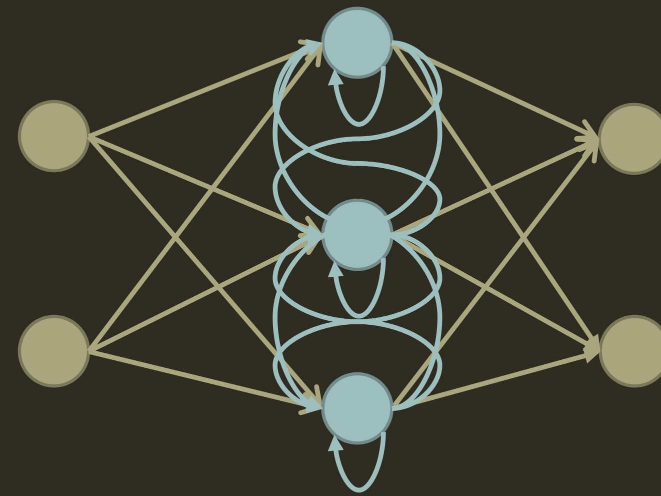
An RNN has feedback connections in its structure so that it 'remembers' n previous inputs, when reading in a sequence.

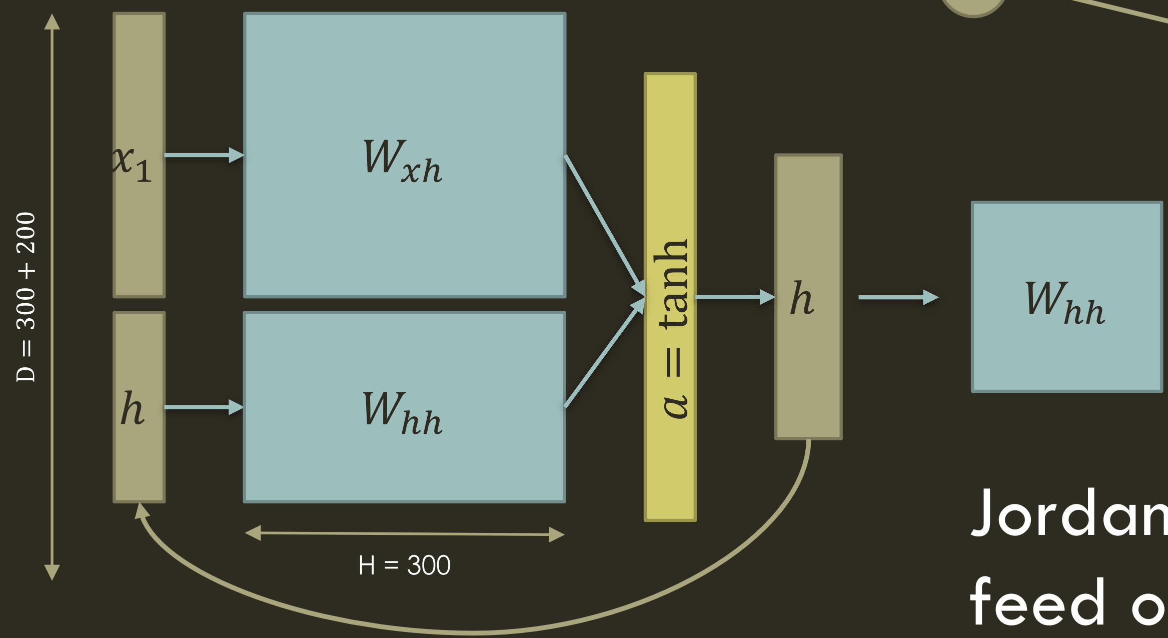
(e.g., can use current word input with hidden units from previous word)



RECURRENT NEURAL NETWORKS (RNNS)

Elman network feed hidden units back

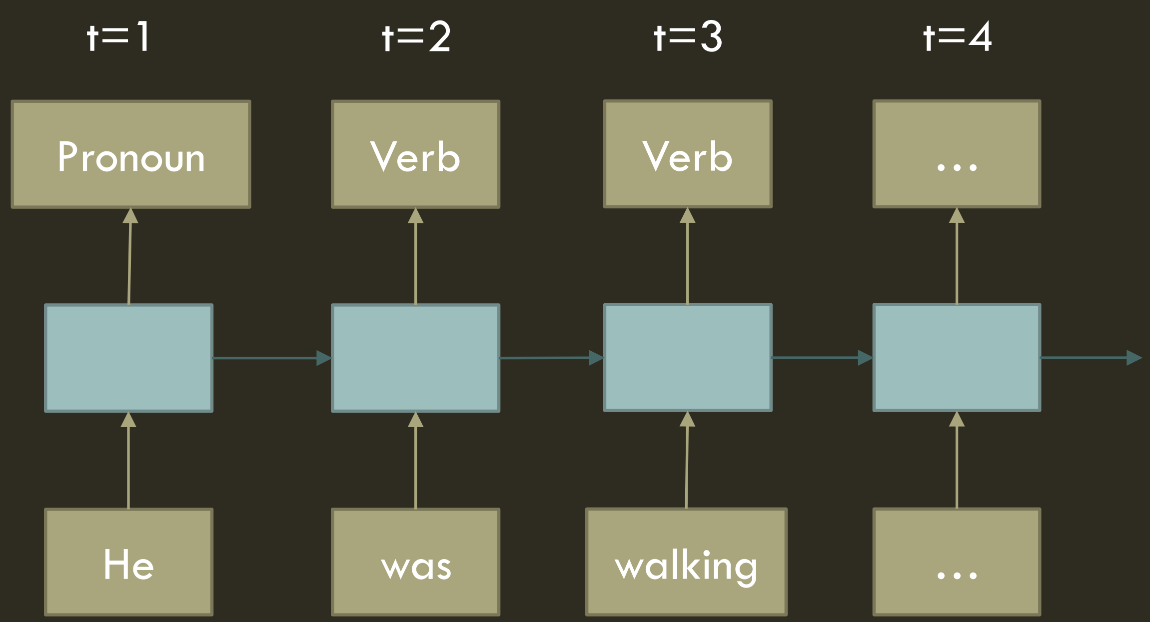




Jordan network (not shown) feed output units back

RNNS ON POS TAGGING

You can 'unroll' RNNs over time for various dynamic models, e.g., PoS tagging.



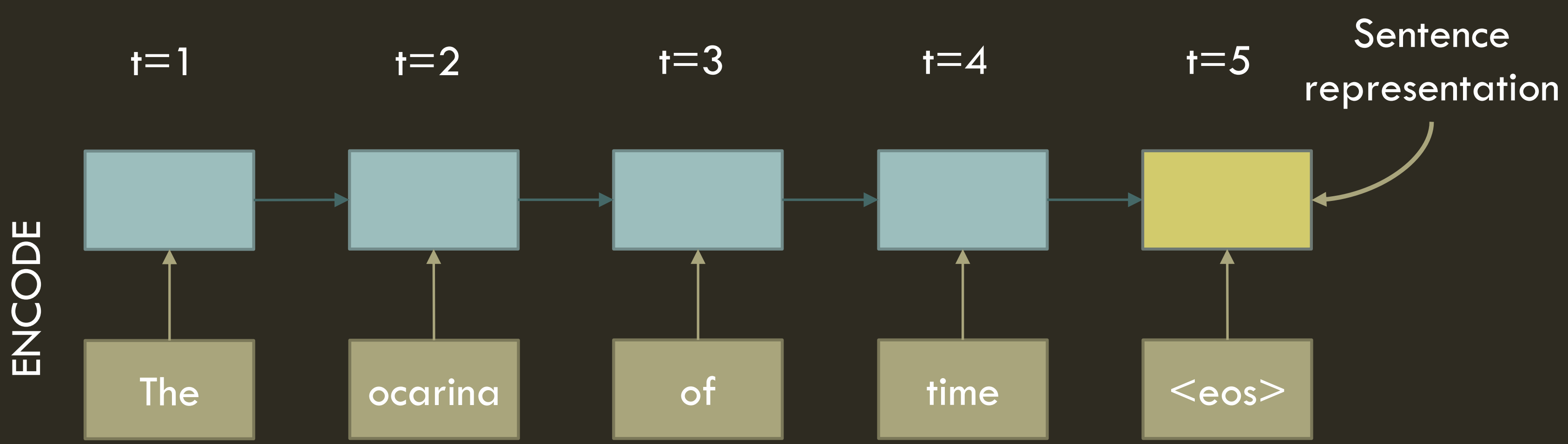
STATISTICAL MACHINE TRANSLATION

SMT is not as easy as PoS.

1. Lexical ambiguity (‘*kill the Queen*’ vs. ‘*kill the queen*’)
2. Different word orders (‘*the blue house*’ vs. ‘*la maison bleu*’)
3. Unpreserved syntax
4. Syntactic ambiguity
5. Idiosyncrasies (‘*estie de sacremouille*’)
6. Different sequence lengths across languages

MACHINE TRANSLATION WITH RNNS

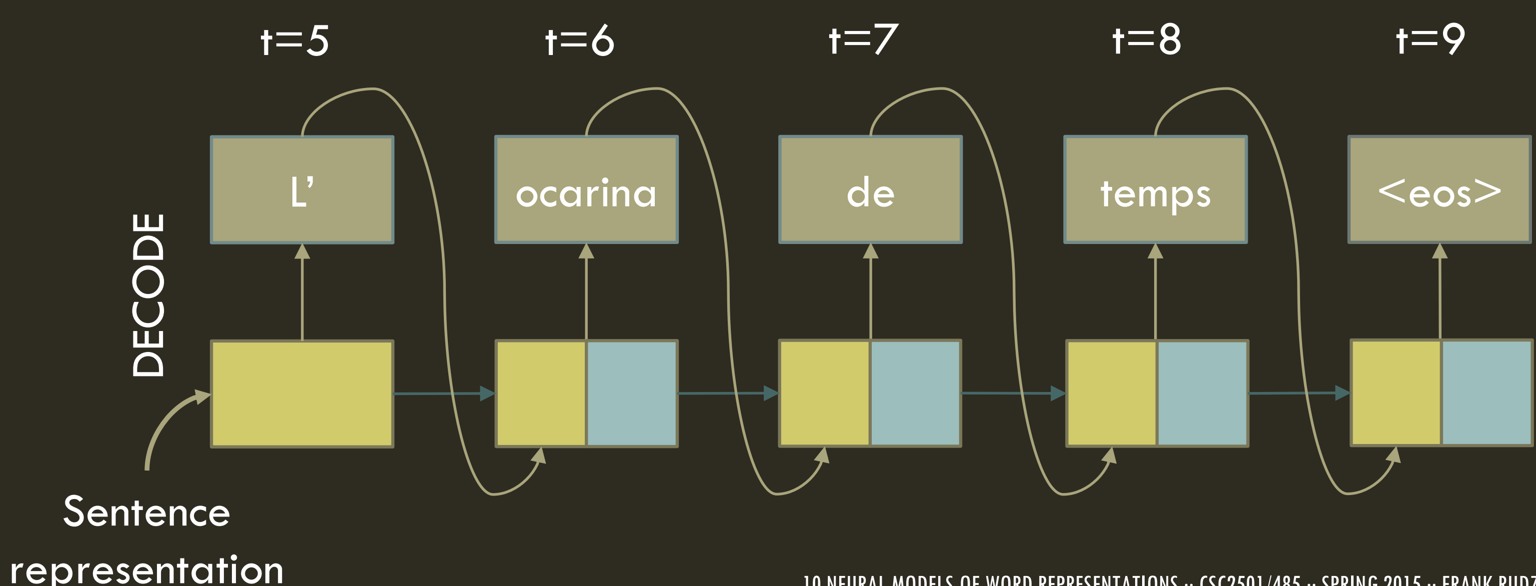
Solution: Encode entire sentence into 1 vector representation, then decode.



MACHINE TRANSLATION WITH RNNS

Try it (http://104.131.78.120/). 30K vocabulary, 500M word training corpus (taking 5 days on GPUs)

- All that good morphological/syntactic/semantic stuff we've seen earlier gets embedded into sentence vectors.



WRAP-UP

‘Negative sampling’:	*n.* contrast random ‘correct’ instances with negative similar examples.
‘skip-gram’:	*n.* the opposite of CBOW; it predicts the context given the centre word rather than the inverse.

With slide material from Yoshua Bengio, Fréderic Godin, Richard Socher, and others (where indicated).