


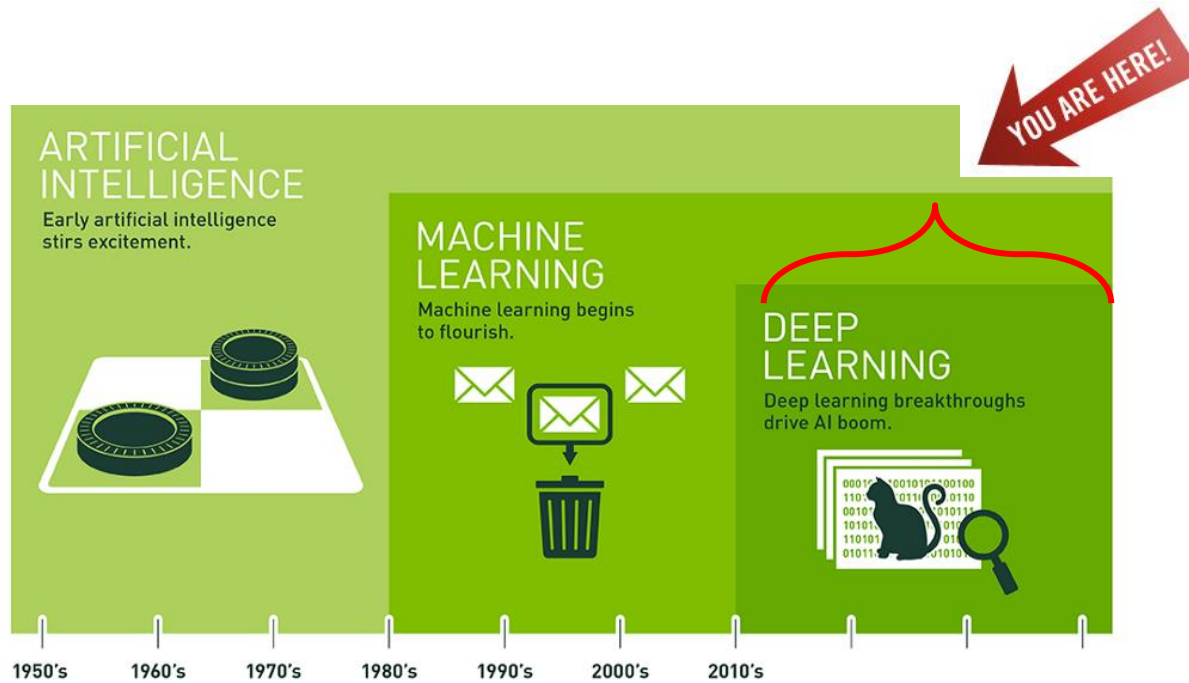
# Artificial Intelligence: Deep Learning for NLP

many slides from: Y. Bengio, A. Ng and Y. LeCun

# Today

1. Introduction 
2. Word Embeddings
3. Deep Learning for NLP

# History of AI



# Deep Learning in the News (2013)



## 10 BREAKTHROUGH TECHNOLOGIES 2013

Introduction

The 10 Technologies

Past Years



### Deep Learning

With massive amounts of computational power, machines can now recognize objects and translate speech in real time. Artificial intelligence is finally getting smart. →

### Temporary Social Media

Messages that quickly self-destruct could enhance the privacy of online communications and make people freer to be spontaneous. →

### Prenatal DNA Sequencing

Reading the DNA of fetuses will be the next frontier of the genomic revolution. But do you really want to know about the genetic problems or musical aptitude of your unborn child? →

### Additive Manufacturing

Skeptical about 3-D printing? GE, the world's largest manufacturer, is on the verge of using the technology to make jet parts. →

### Baxter: The Blue-Collar Robot

Rodney Brooks's newest creation is easy to interact with, but the complex innovations behind the robot show just how hard it is to get along with people. →

### Memory Implants

A maverick neuroscientist believes he has deciphered the code by which the brain forms long-term memories. Next: testing a prosthetic implant for people suffering from long-term memory loss. →

### Smart Watches

The designers of the Pebble watch realized that a mobile phone is more useful if you don't have to take it out of your pocket. →

### Ultra-Efficient Solar Power

Doubling the efficiency of a solar cell would completely change the economics of renewable energy. Nanotechnology just might make it possible. →

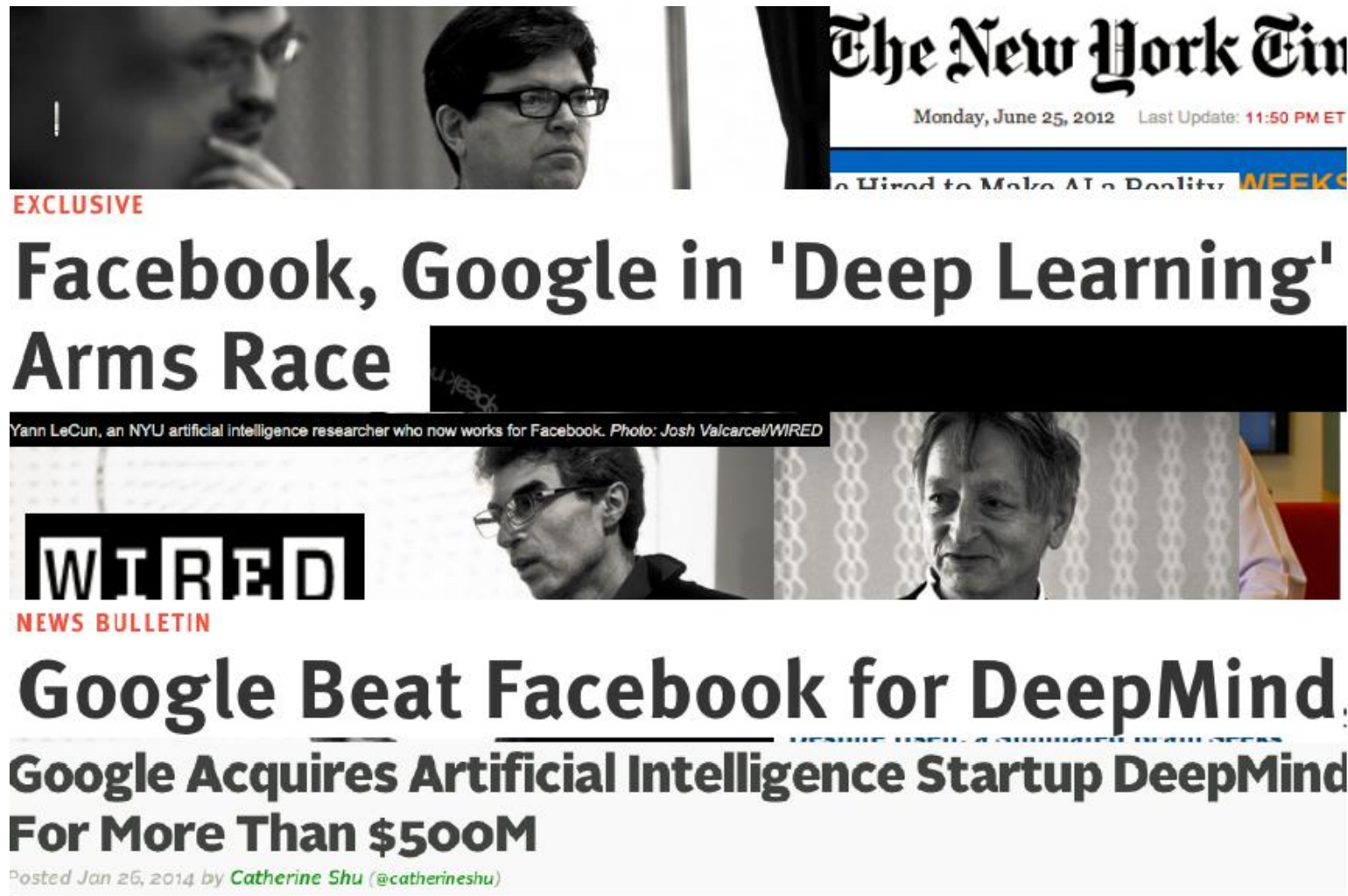
### Big Data from Cheap Phones

Collecting and analyzing information from simple cell phones can provide surprising insights into how people move about and behave – and even help us understand the spread of diseases. →

### Supergrids

A new high-power circuit breaker could finally make highly efficient DC power grids practical. →

# Deep Learning in the News (2012-2014)



# Major Breakthroughs

- Speech Recognition & Machine Translation (2010+)
- Image Recognition & Computer Vision (2012+)
- **Natural Language Processing (2014+)**

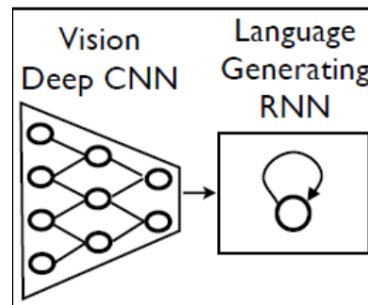
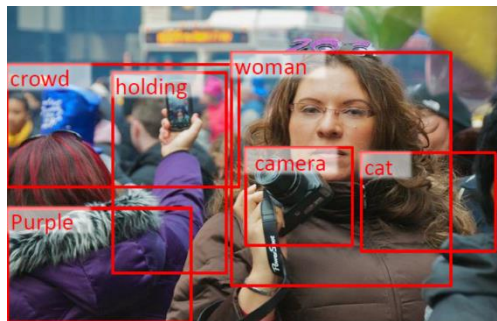
Joe went to the kitchen. Fred went to the kitchen. Joe picked up the milk.  
Joe travelled to the office. Joe left the milk. Joe went to the bathroom.

Where is the milk now? A: office

Where is Joe? A: bathroom

Where was Joe before the office? A: kitchen [http://blog.csdn.net/qfnu\\_cjt\\_wl](http://blog.csdn.net/qfnu_cjt_wl)

*Question Answering*



Machine-generated (but turker preferred)

a bicycle is parked next to a river

Human-annotated (but turker not preferred)

a bike sits parked next to a body of water

*Image Captioning (deep vision + deep NLP)*



# Image Captioning: Better than humans?



**A**

a woman in a kitchen preparing food

**B**

woman working on counter near kitchen sink preparing a meal

# Many Types of Neural Networks

A mostly complete chart of

## Neural Networks

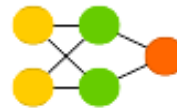
©2016 Fjodor van Veen - asimovinstitute.org

- Backfed Input Cell
- Input Cell
- △ Noisy Input Cell
- Hidden Cell
- Probabilistic Hidden Cell
- △ Spiking Hidden Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- △ Different Memory Cell
- Kernel
- Convolution or Pool

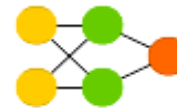
Perceptron (P)



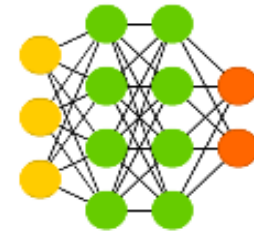
Feed Forward (FF)



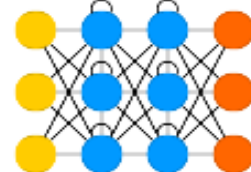
Radial Basis Network (RBF)



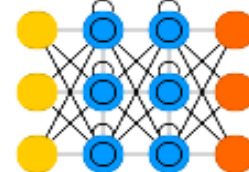
Deep Feed Forward (DFF)



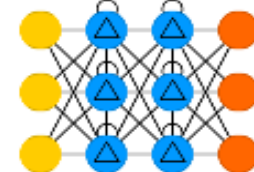
Recurrent Neural Network (RNN)



Long / Short Term Memory (LSTM)



Gated Recurrent Unit (GRU)



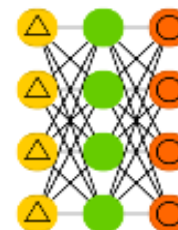
Auto Encoder (AE)



Variational AE (VAE)



Denoising AE (DAE)

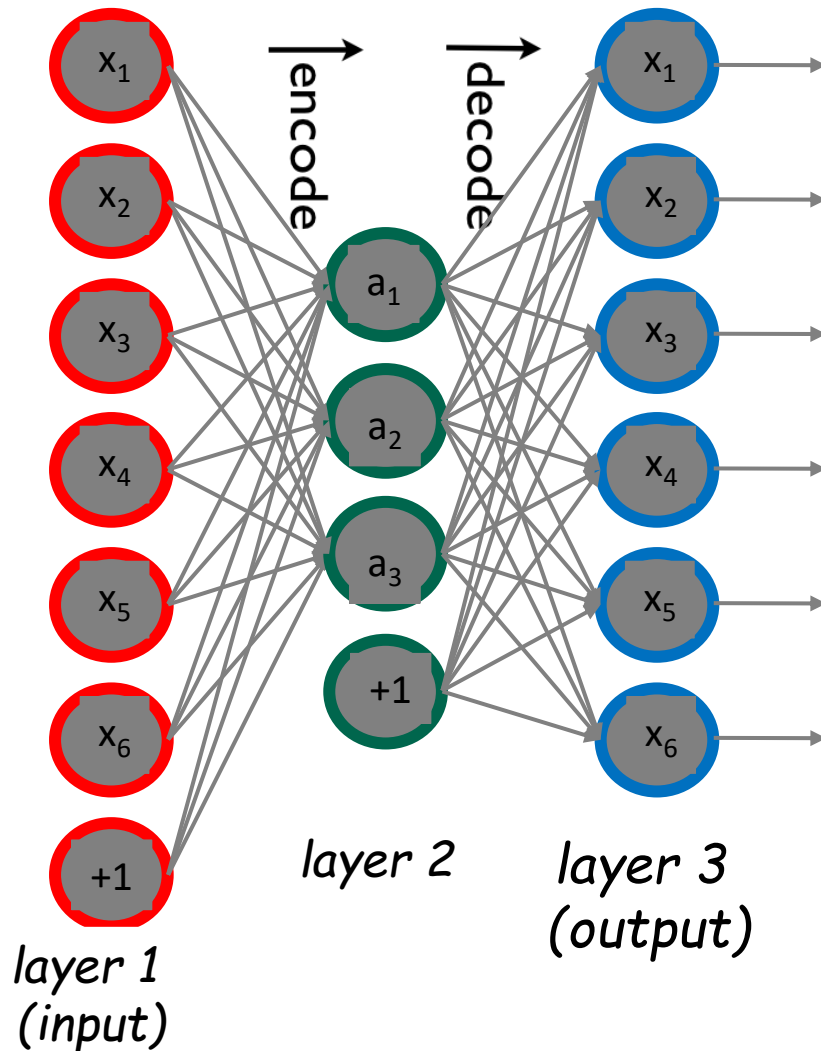


Sparse AE (SAE)



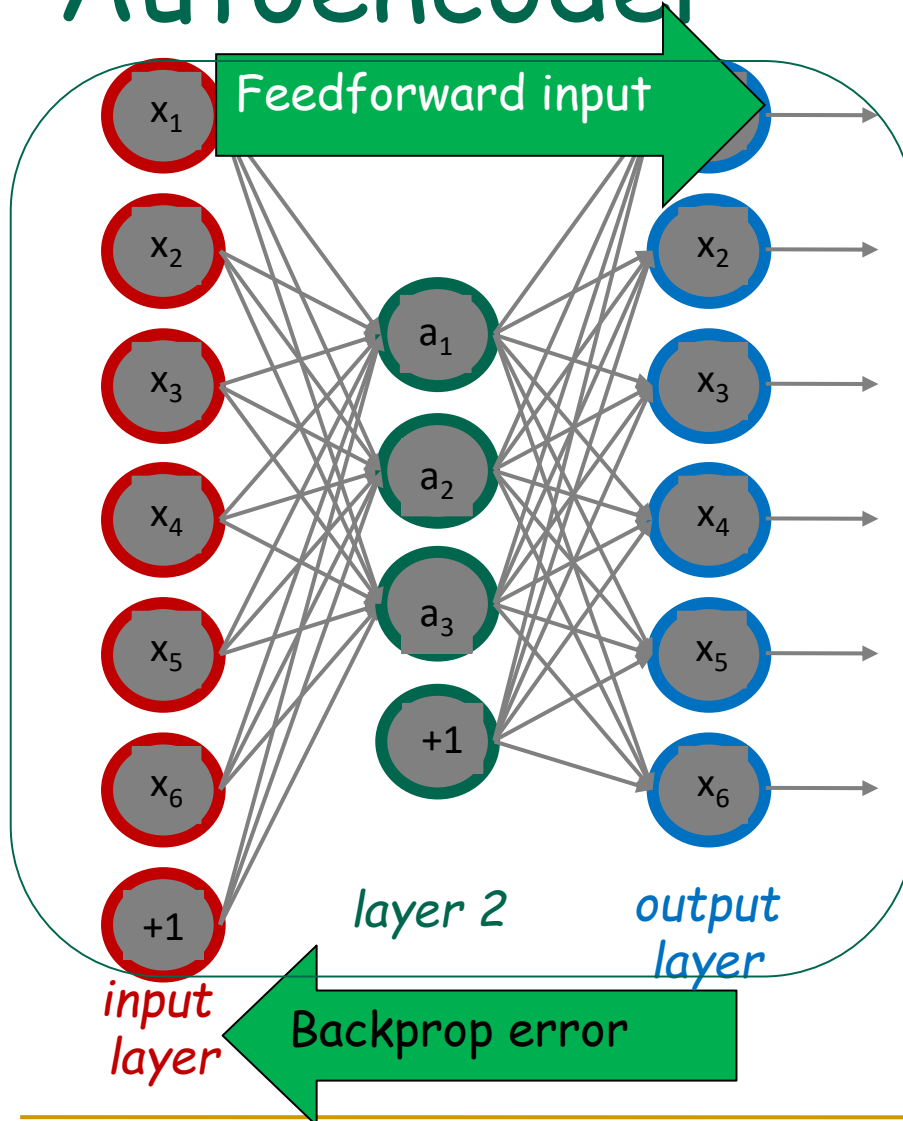


# Autoencoder

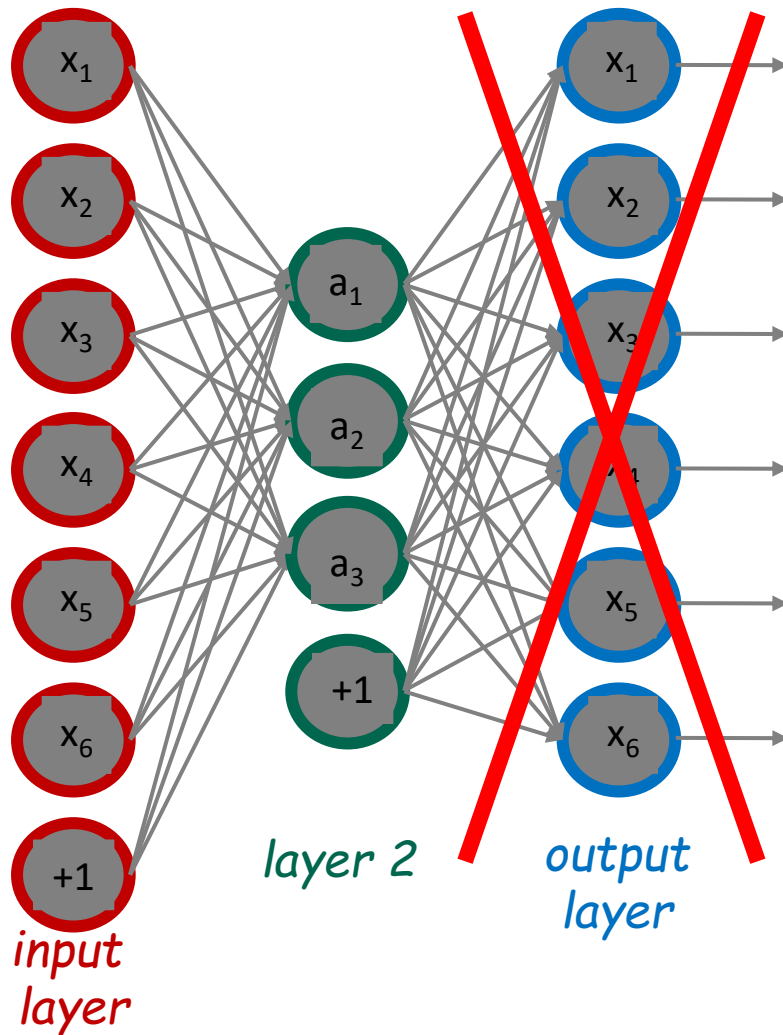


- The network is trained to output the input i.e. learn the identity function.
- Trivial... unless, we impose constraints:
  - Nb of units in layer 2 < nb of input units (learn compressed representation)  
OR
  - Constrain layer 2 to be sparse (i.e. many connections are "disabled")

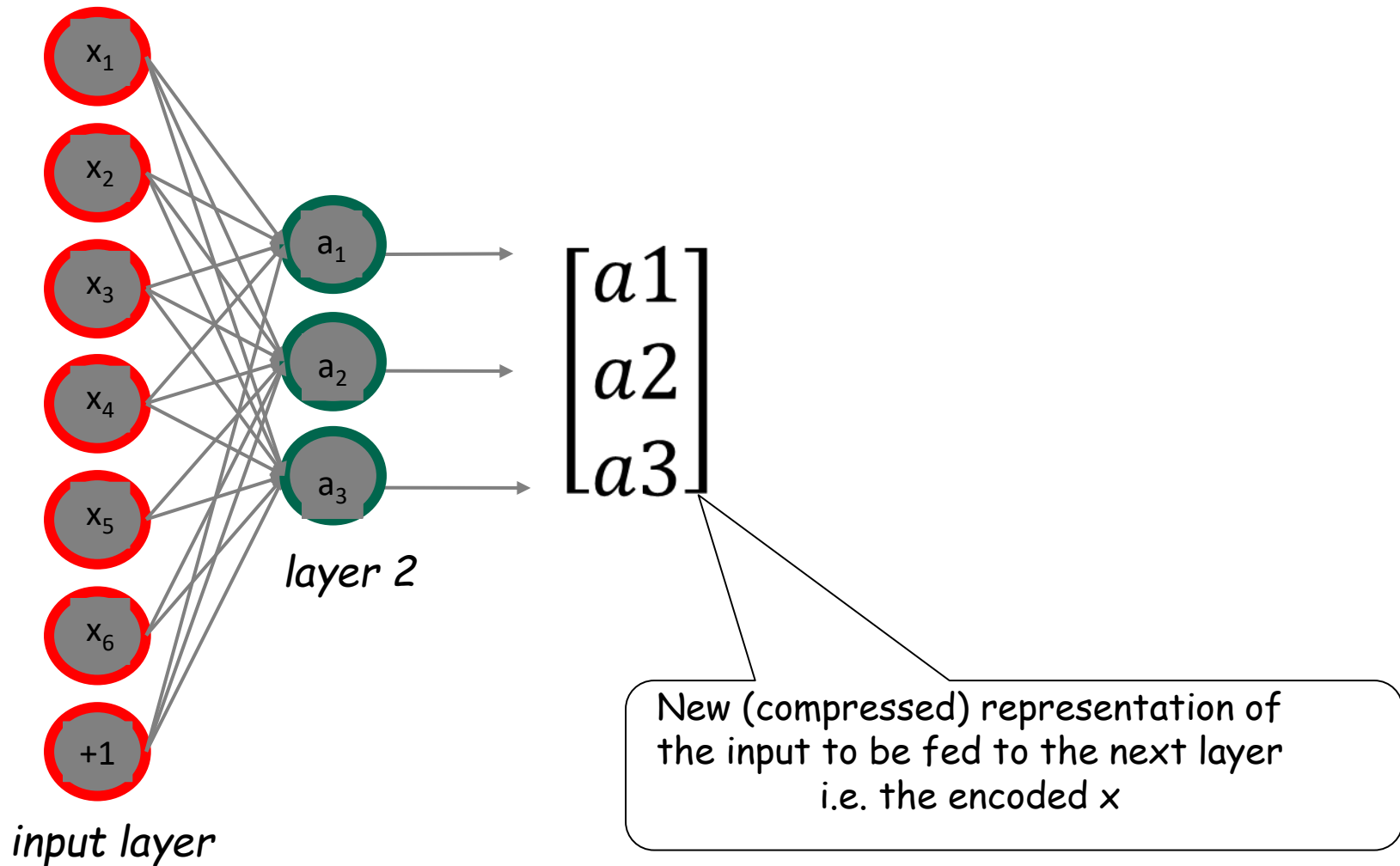
# Autoencoder



# Autoencoder



# Autoencoder



# Today

1. Motivation
2. Word Embeddings
3. Deep Learning for NLP



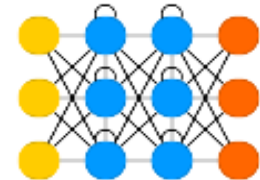
# Deep Learning for NLP

Deep learning models for NLP use

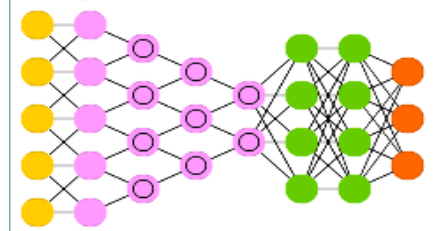
- Vector representation of words
  - i.e., word embeddings
- Neural network structures
  - Recurrent Neural Networks (RNNs)
  - Convolutional Networks (CNNs)
  - Recursive Neural Networks
  - ...



Recurrent Neural Network (RNN)



Deep Convolutional Network (DCN)





# Word Embeddings

- To do NLP with neural networks, words need to be represented as vectors
- Traditional approach: "one hot vector"
  - Binary vector
  - Length = | vocab |
  - 1 in the position of the word id, the rest are 0
  - [0, 0, 0, 1, 0, 0, 0, ...]
- However, this does not represent word meaning :-)
- Similar words such as *python* and *ruby* should have similar vector representations
- However, similarity/distance between all "one hot vectors" is the same

*python*

0	1	0	0	0
---	---	---	---	---

*ruby*

0	0	0	1	0
---	---	---	---	---

*python*

0	1	0	0	0
---	---	---	---	---

⊗

*ruby*

0	0	0	1	0
---	---	---	---	---

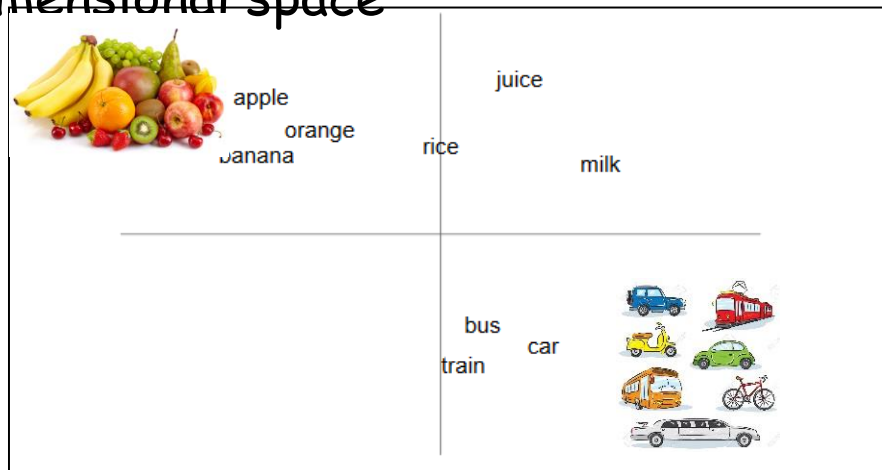
||

0	0	0	0	0
---	---	---	---	---

# Word Embeddings

- We would like:
  - `cat/kitty/dog` ... to have similar representations
  - `cat/orange/train/python`... to have dissimilar representations
- Word embeddings:
  - aka. word representations
  - Represent each word by a vector of fixed dimensions (eg,  $n = 50$  to  $300$ )
  - Like a point in  $n$ -dimensional space


	50~300 dim			
python	0.52	0.21	0.37	...
ruby	0.48	0.21	0.33	...
word	0.05	0.23	0.06	...



# Word2vec

- Popular embedding method
- Very fast to train
- Code available on the web
  - <https://code.google.com/archive/p/word2vec>
- Idea:
  - **predict** rather than **count**
  - use unsupervised texts from the Web

# Word2vec

- Instead of **counting** how often each word  $w$  occurs near "apricot"
- Train a classifier on a binary **prediction** task:
  - Is  $w$  likely to show up near "cat"?
- In the end, we do not actually care about this task 
  - But we will take the learned weights as the word embeddings
- Use as training set readily available texts, so no need for hand-labeled supervision !

# Word2vec Models

*"A word is known by the company it keeps"* – J. R. Firth

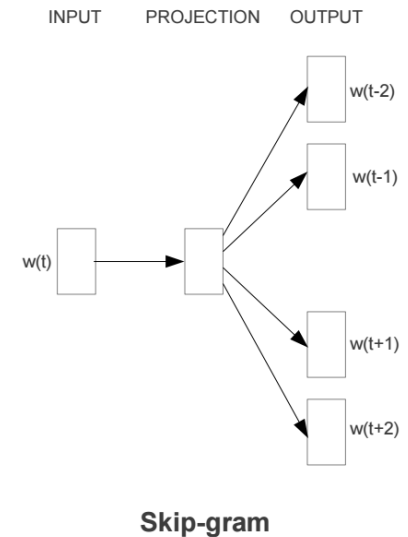
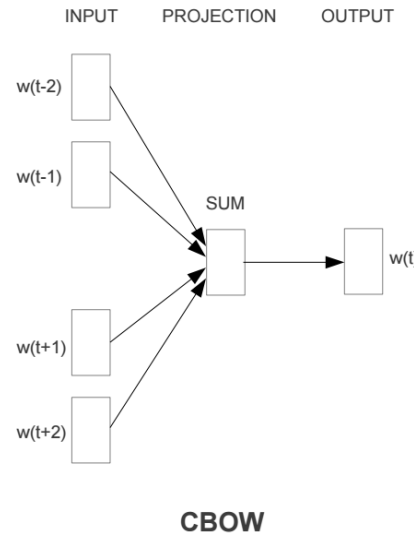
- Basic Idea:

1. Similar words should have similar contexts (surrounding words)
2. So we can use the contexts to guess the word (or vice-versa).
  - The cat/kitty/dog hunts for mice.
  - The brown furry cat/kitty/dog is eating.
  - John's cat/kitty purrs.
3. Train an ANN to guess a word given its context (or vice-versa)

# Word2Vec models

Word2Vec has 2 models:

1. **CBOW**: given context words, guess the word
2. **Skip-gram**: given a word, guess one of its surrounding word





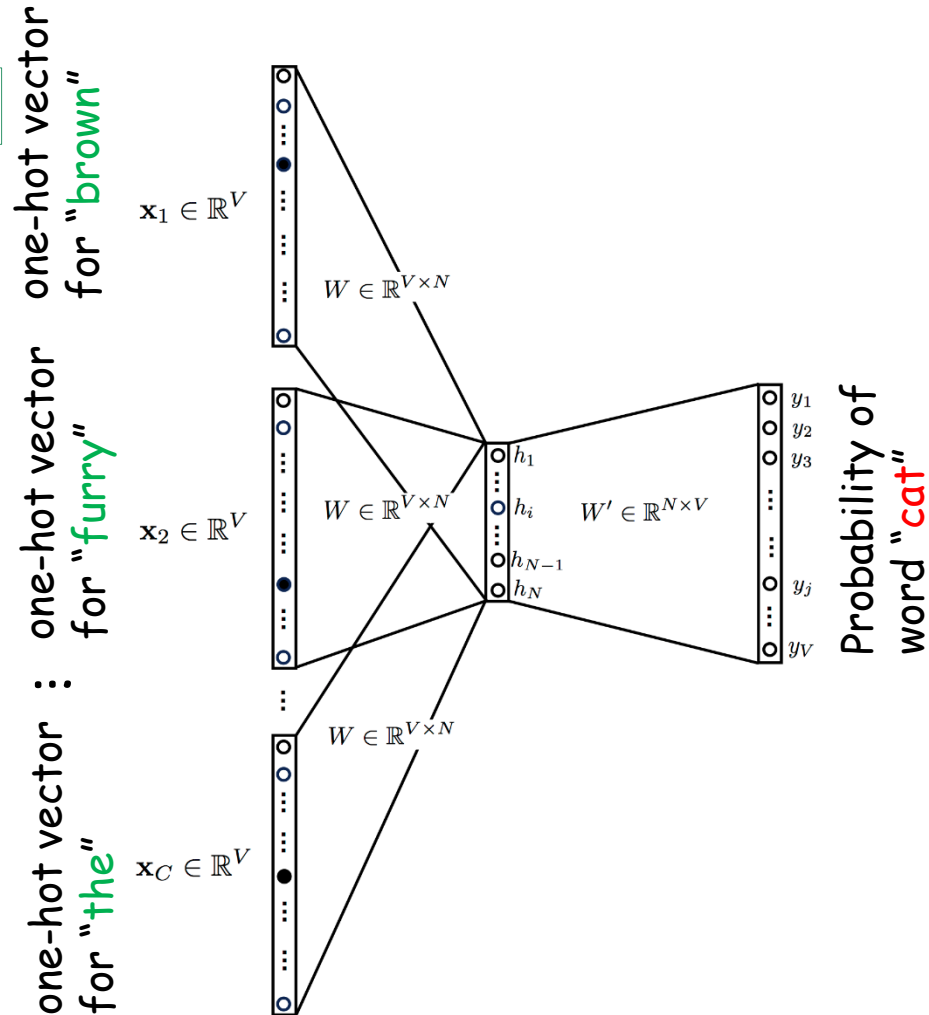
# Word2Vec: CBOW Model

The brown furry cat chases the mouse

Uses a shallow neural network with only 3 layers:

1. One input layer
2. One hidden layer and
3. One output layer.

goal: predict the probability of a target word (cat) given a context (brown furry chases the).



# Word2Vec - Creating the Data Set

The brown furry cat chases the mouse inside the house.

...

...

Assume context words are those in +/- 2 word window

Instance	Context word -2	Context word -1	Context word +1	Context word +2	To Predict
1	the	brown	cat	chases	furry
2	brown	furry	chases	the	cat
3	furry	cat	the	mouse	chases
4	cat	chases	mouse	inside	the
5	chases	the	inside	the	mouse
6	the	mouse	the	house	inside

# Word2Vec - Input to the Network

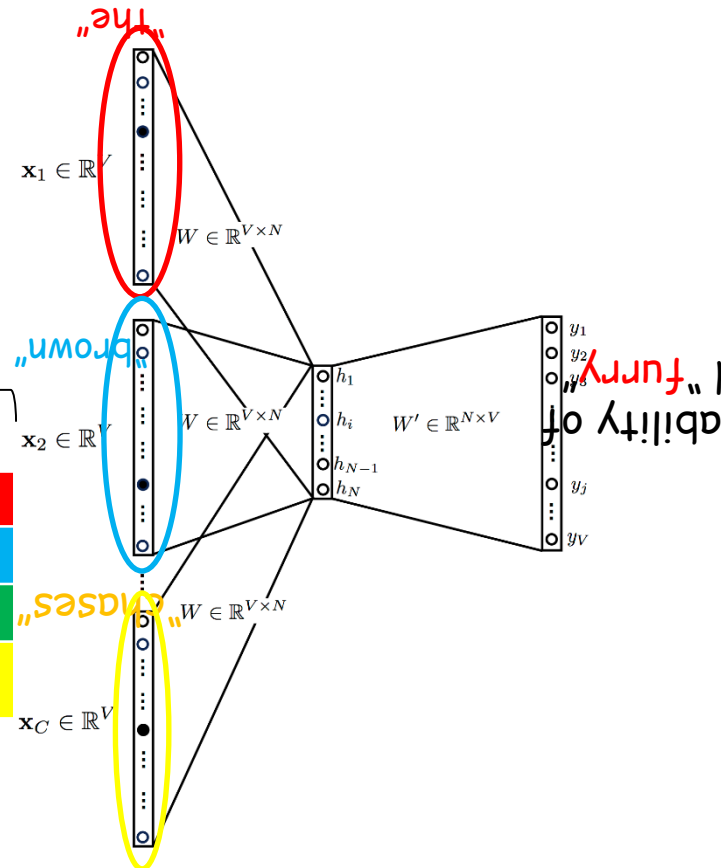
$V$  = size of vocabulary

$N$  = size of the embedding that we want  
(i.e. number of neurons in the hidden layer)

$C$  = size of context (2 words before + 2 after)

Inst ance	Input word		$V$									
1	Context -2	the	0	1	0	0	0	0	0	0	0	...
1	Context -1	brown	0	0	0	0	0	1	0	0	0	...
1	Context +1	cat	1	0	0	0	0	0	0	0	0	...
1	Context +2	chases	0	0	0	1	0	0	0	0	0	...

Inst ance	Input word		$V$									
2	Context -2	brown	0	0	0	0	0	1	0	0	0	...
2	Context -1	furry	0	0	0	0	1	0	0	0	0	...
2	Context +1	chases	0	0	0	0	0	0	0	0	1	...
...	Context +2	the	0	1	0	0	0	0	0	0	0	...



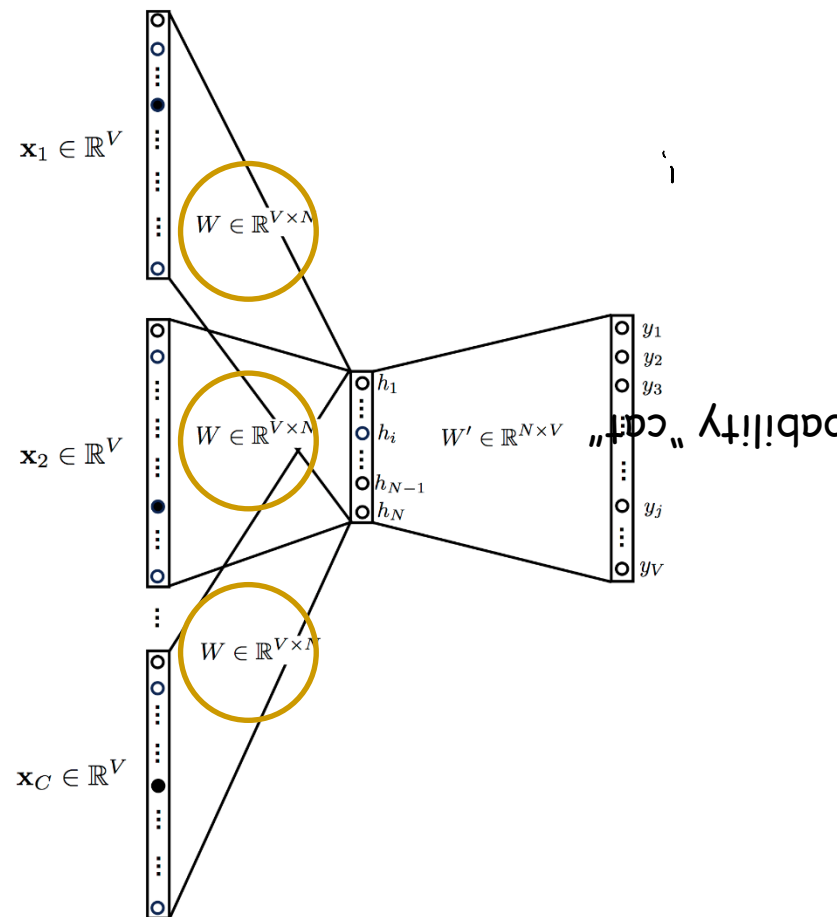
# Word2Vec - Weights W

- Weight Matrix  $W$  between input & hidden layer
- $W$  is a  $V \times N$  matrix...
- Initially random but modified via backprop
- No bias!

$V$

1	2	3	4	5	...
6	7	8	9	10	...
11	12	13	14	15	...
16	17	18	19	20	...
21	22	23	24	25	...
26	27	28	29	30	...
31	32	33	34	35	...
36	37	38	39	40	...
41	42	43	44	45	...
46	47	48	49	50	...
...	...	...	...	...	...

$N$  (nb of nodes in hidden layer, i.e. size of the embedding)



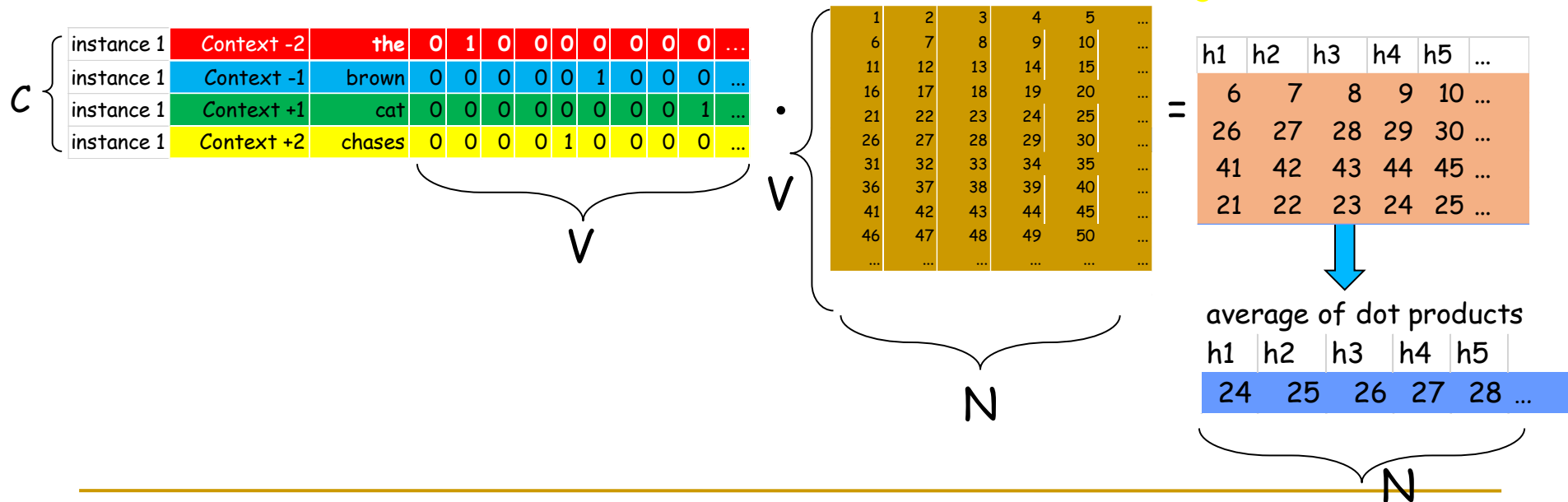
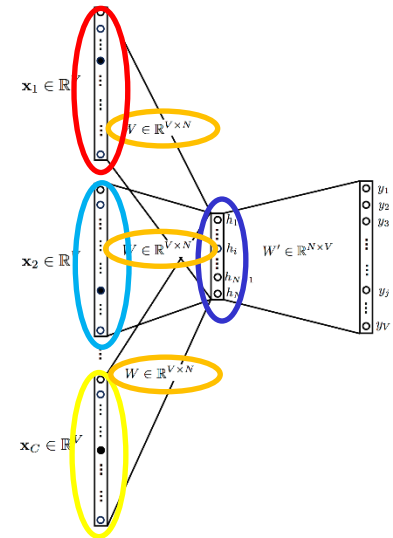
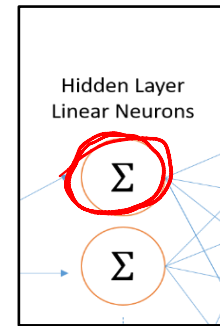
# Word2Vec - Feedforward

1. Calculate the output of each of the  $N$  hidden nodes for each context word

Note: there is **no activation function**.

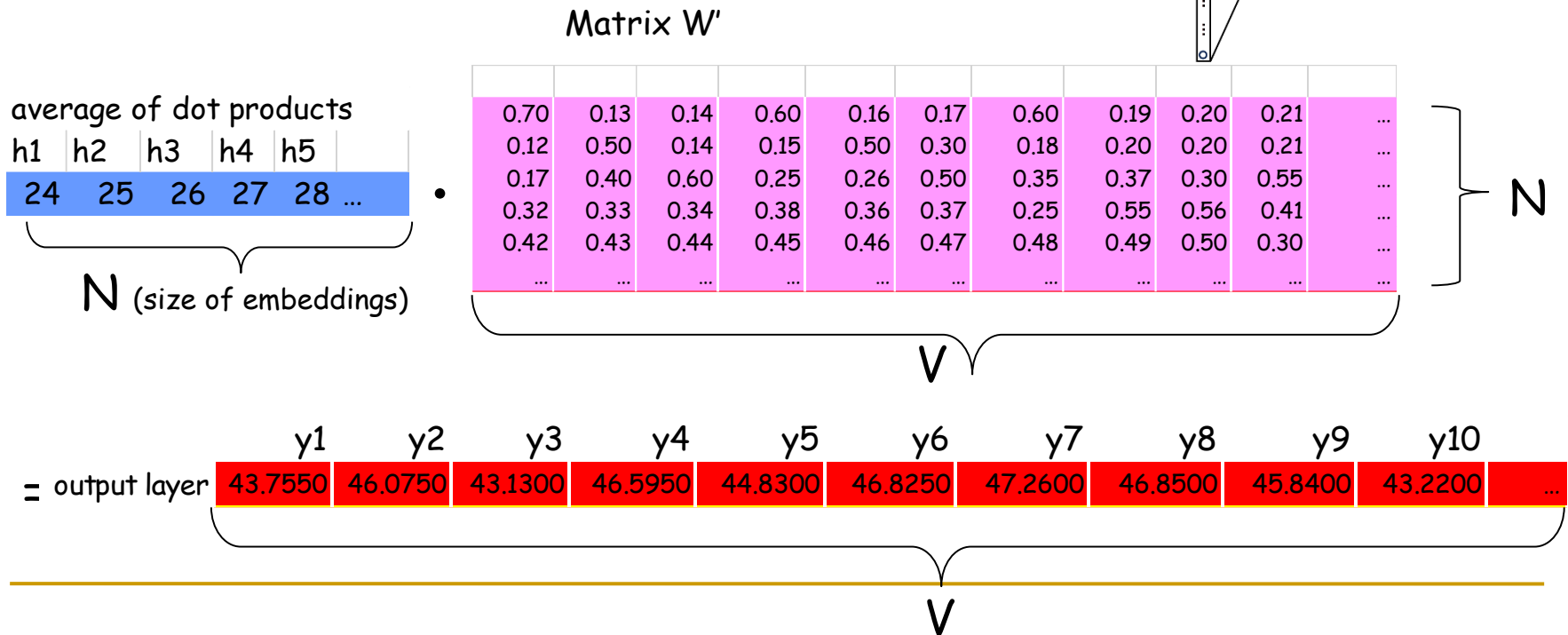
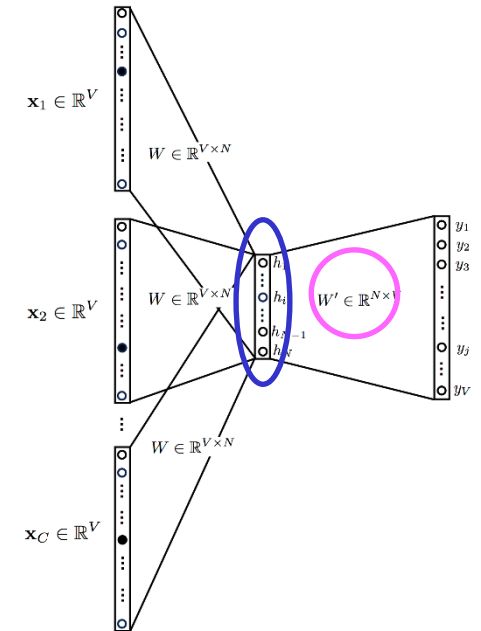
Output of  $h_i$  is just the **dot product**

2. Then take the **average**



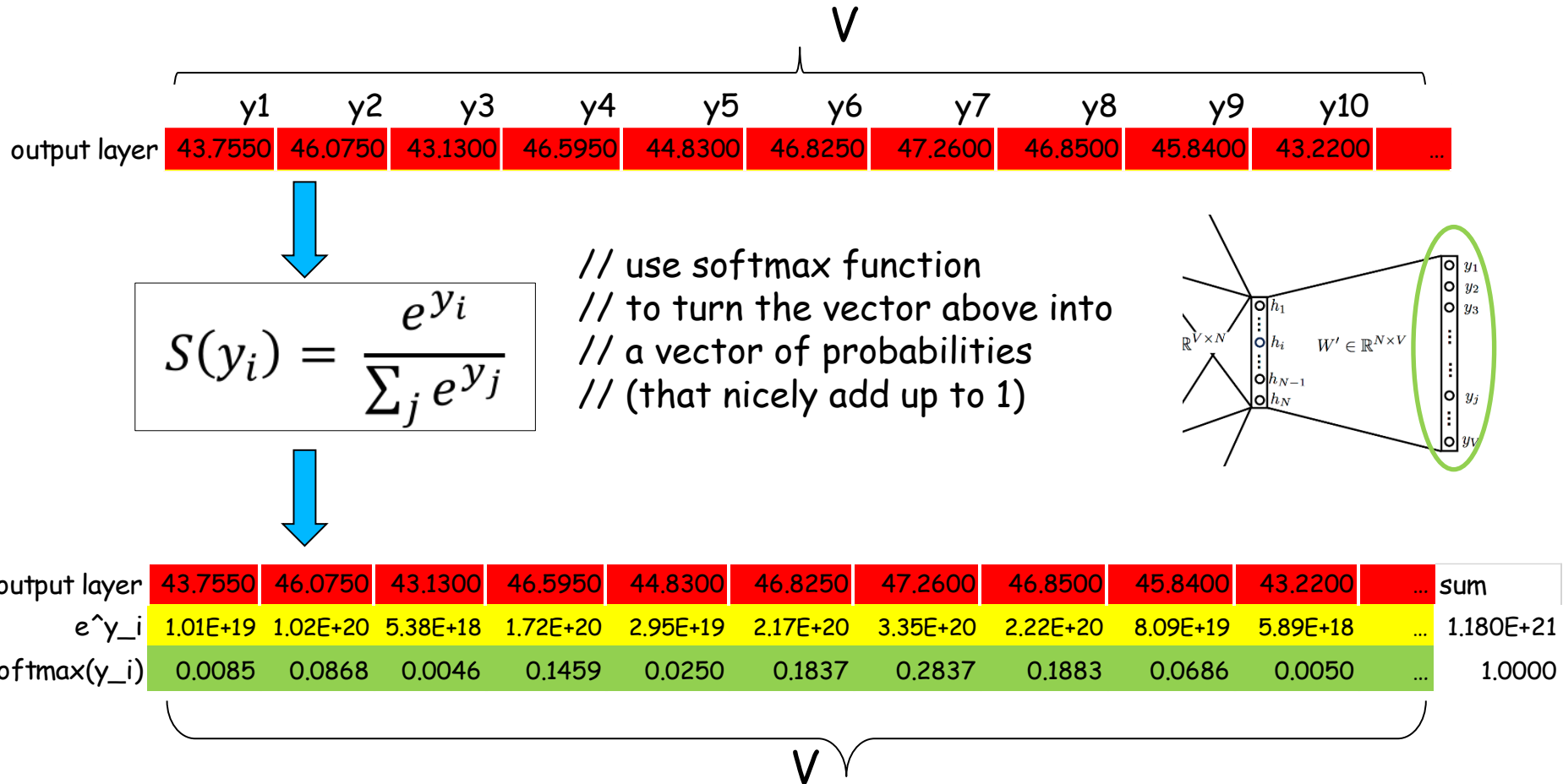
# Word2Vec - Weights $W'$

- Weight Matrix  $W'$  between hidden & output layer
- $W'$  is a  $N \times V$  matrix...
- Initially random but modified via backprop
- Feed forward **average of hidden neurons** and do dot product with **matrix  $W'$**





# Turn dot product into probabilities



# Compute Error of output layer

remember the training set:

Instance	Context word -2	Context word -1	Context word +1	Context word +2	To Predict
1	the	brown	cat	chases	furry
2	brown	furry	chases	the	cat
...	...	...	...	...	...

// target = 1 hot representation of the target (furry) in the training set

	V									
	y1	y2	y3	y4	y5	y6	y7	y8	y9	y10
O= output	0.0085	0.0868	0.0046	0.1459	0.0250	0.1837	0.2837	0.1883	0.0686	0.0050 ...
-										
T= target	t1	t2	t3	t4	t5	t6	t7	t8	t9	t10
	0	0	0	0	0	0	1	0	0	0 ...
=										
O-T	0.0085	0.0868	0.0046	0.1459	0.0250	0.1837	-0.7163	0.1883	0.0686	0.0050 ...

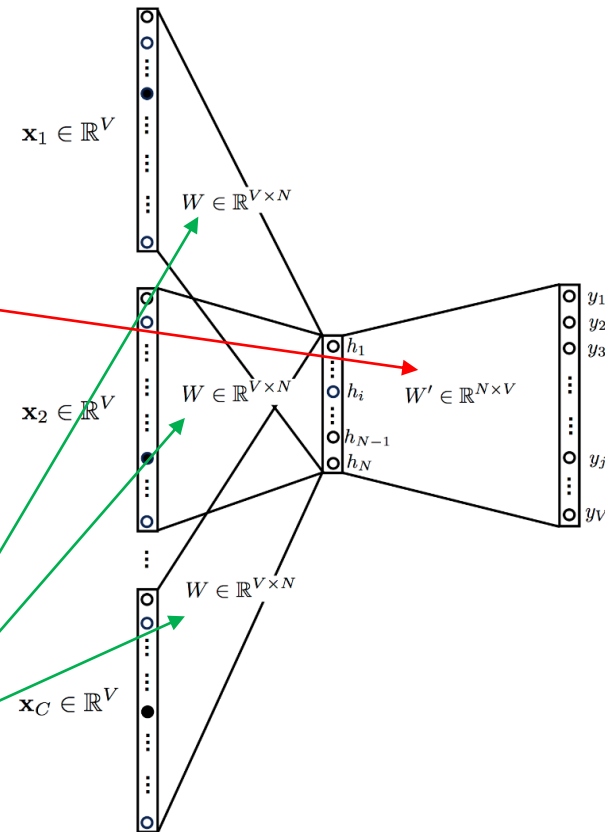
# Backpropagate errors to adjust $W$ and $W'$

- Adjust  $W'$  and  $W$  using backpropagation
- <after a bit of math>, we get:

$$w'_{jk} = w'_{jk} - \eta(y_k - t_k)h_j$$

update only the  $w_{ij}$  for the inputs where  $x_1 = 1$

$$w_{ij} = w_{ij} - \eta \frac{1}{C} \sum_{k=1}^V (y_k - t_k) w'_{jk}$$



$\eta$ : learning rate

$C$ : size of context (eg 4)

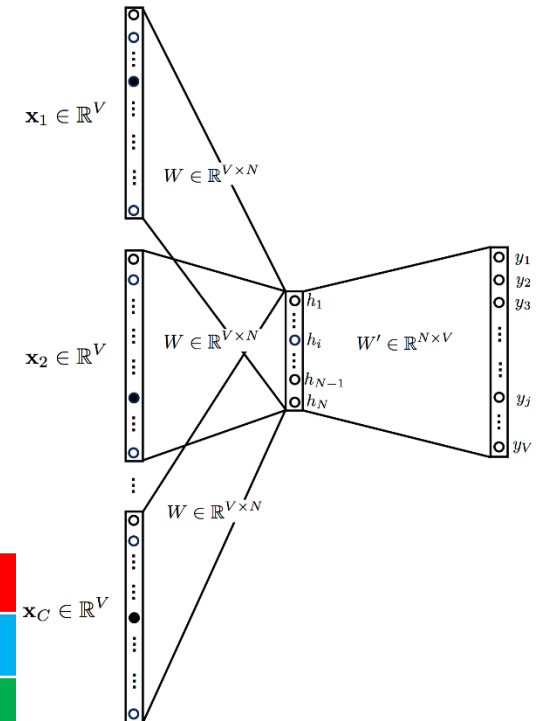
The weight updates are ONLY done on the "rows" of  $W$  that correspond to the input word, not for all elements of  $W$ . Remember that the input words are represented as 1-hot vectors, so only the weights of the word that has a "1" are updated. This makes intuitive sense, as we want to update the weight only of the context word since its previous weights lead to an error.

# Word2Vec - FeedForward next data

Instance	Context word -2	Context word -1	Context word +1	Context word +2	To Predict
1	the	brown	eat	chases	furrry
2	brown	furrry	chases	the	cat
3	furrry	cat	the	mouse	chases
4	cat	chases	mouse	inside	the
5	chases	the	inside	the	mouse
6	the	mouse	the	house	inside

Insta nce    Input word

2	Context -2	brown	0	0	0	0	0	1	0	0	0	...
2	Context -1	furrry	0	0	0	0	1	0	0	0	0	...
2	Context +1	chases	0	0	0	0	0	0	0	0	1	...
1	Context +2	the	0	1	0	0	0	0	0	0	0	...



- Iterate feedforward/ backprop until error is minimized
- Trained on Google News dataset (about 100 billion words).
- See: <https://code.google.com/archive/p/word2vec/>

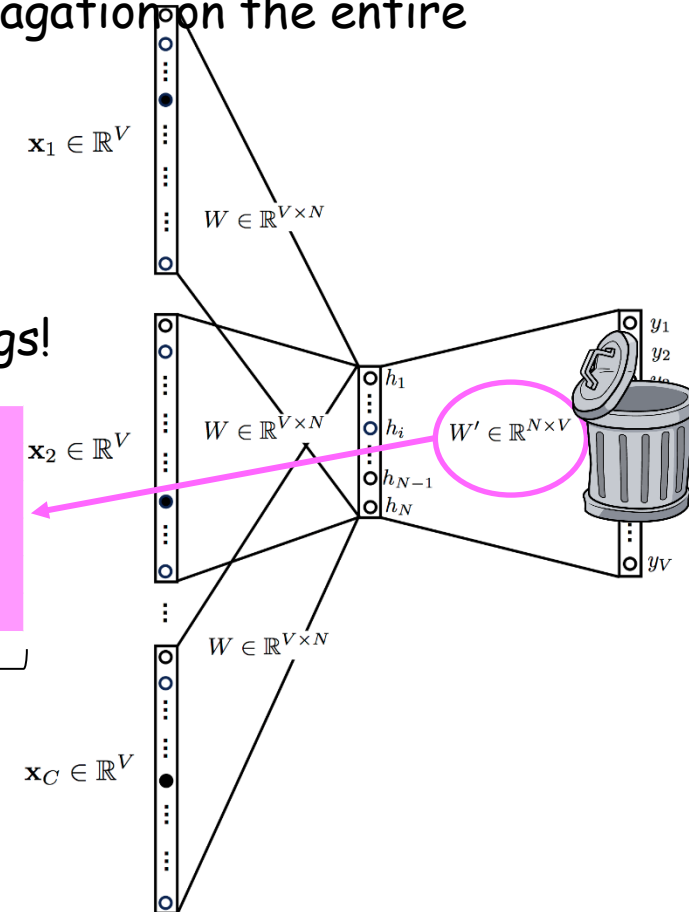
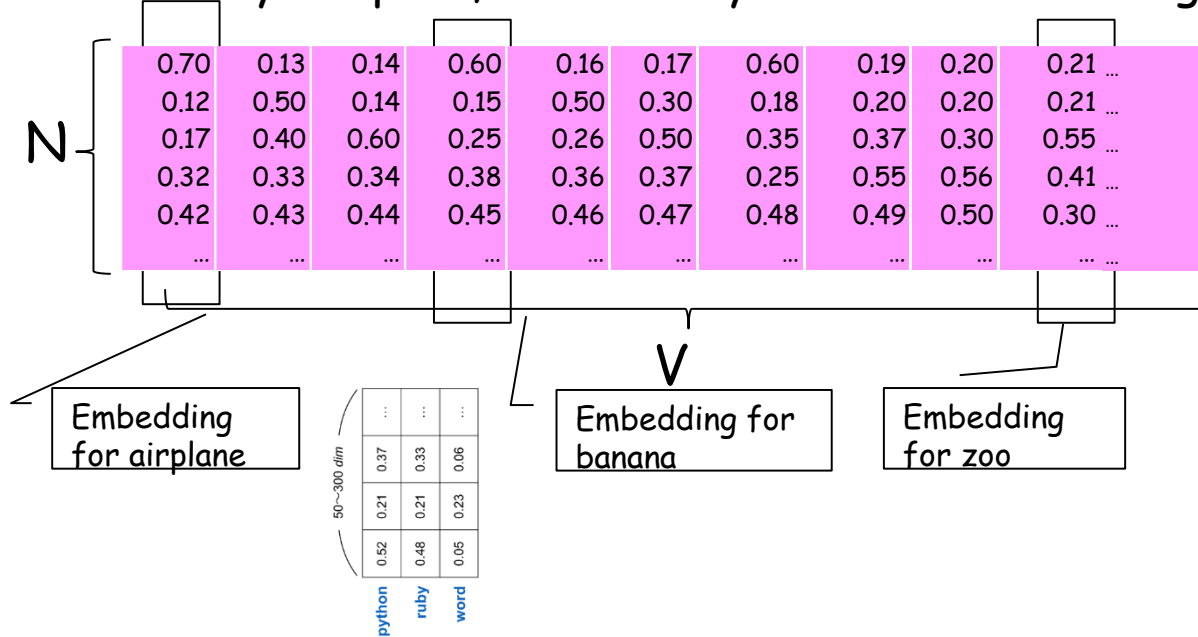


almost...

remember, we did all this to get embeddings...  
I'm not leaving 'till I get my embeddings!

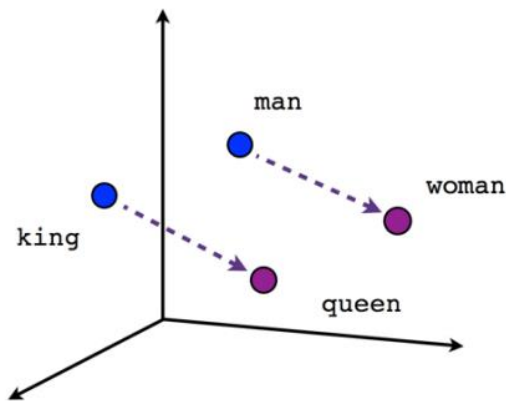
# Word2Vec- Get the embeddings

- After many iterations of feedforward, backpropagation on the entire training set ...
- The classifier will be built!
- Then, we throw it away ! (yes, we do!)
- and only keep  $W'$ , these are your word embeddings!

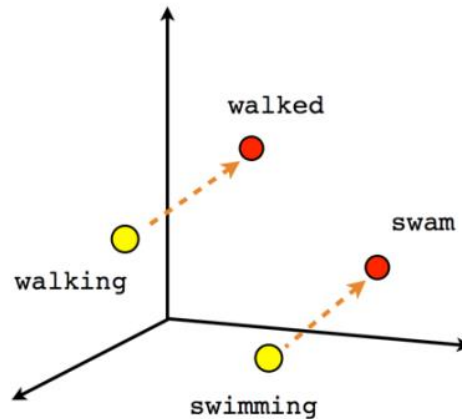




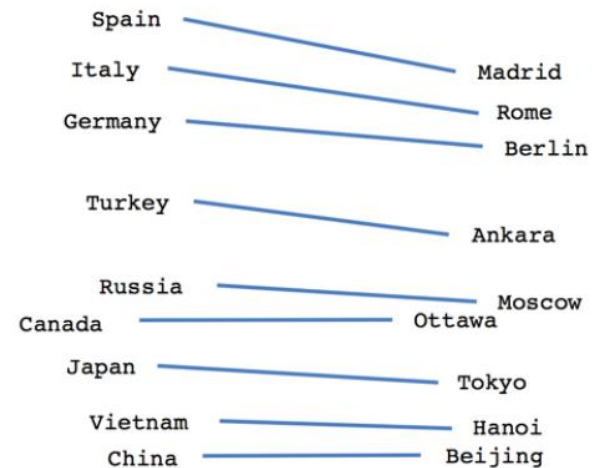
# Results



Male-Female



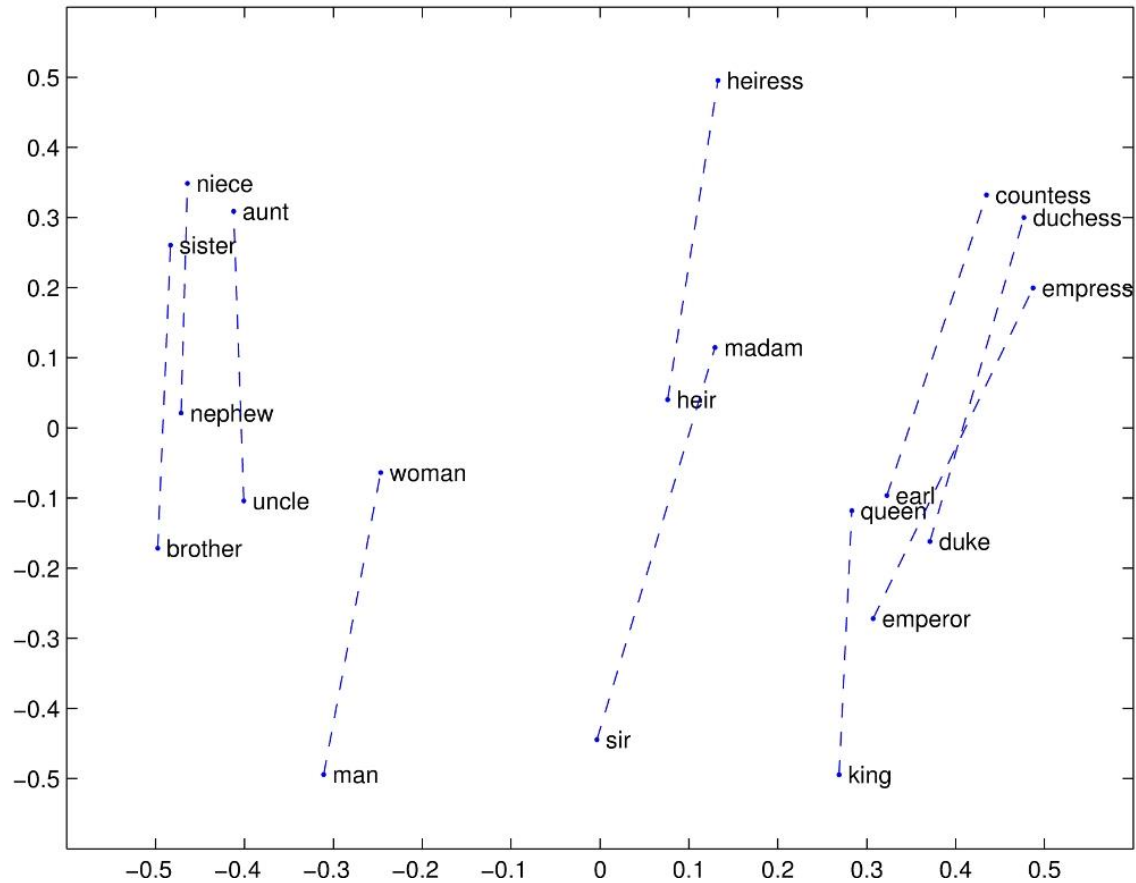
Verb tense



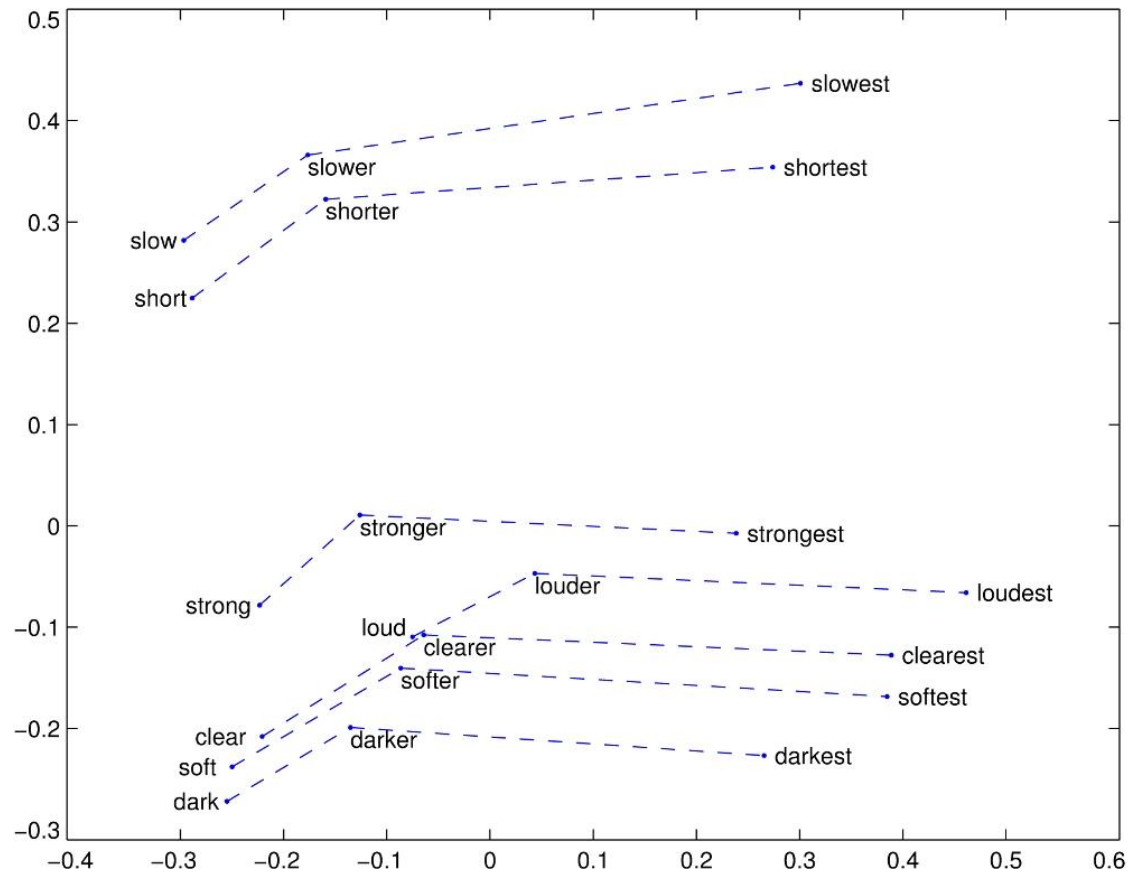
Country-Capital

$\text{vector}[\text{queen}] \sim \text{vector}[\text{king}] - \text{vector}[\text{man}] + \text{vector}[\text{woman}]$   
 $\text{vector}[\text{swimming}] \sim \text{vector}[\text{swam}] - \text{vector}[\text{walked}] + \text{vector}[\text{walking}]$   
 $\text{vector}[\text{Rome}] \sim \text{vector}[\text{Madrid}] - \text{vector}[\text{Spain}] + \text{vector}[\text{Italy}]$

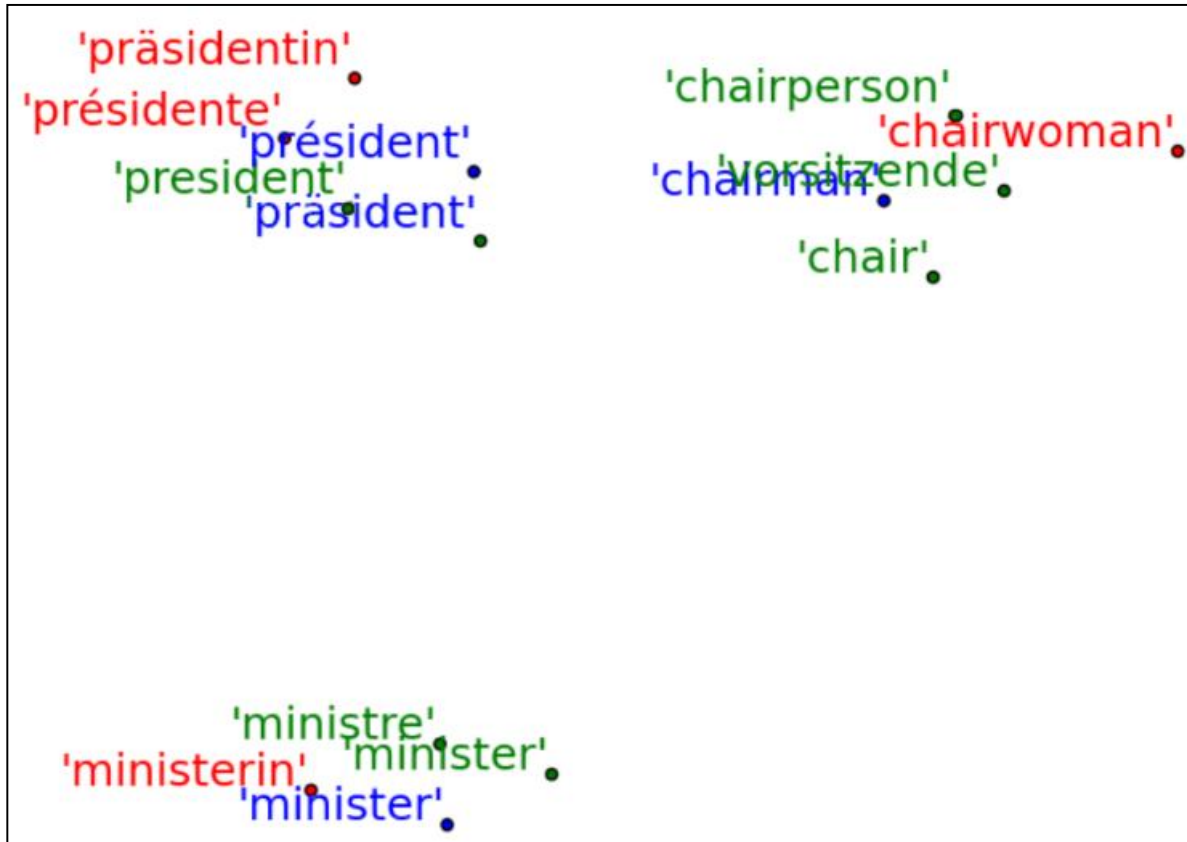
# Results



# Results



# Multilingual Word Embeddings

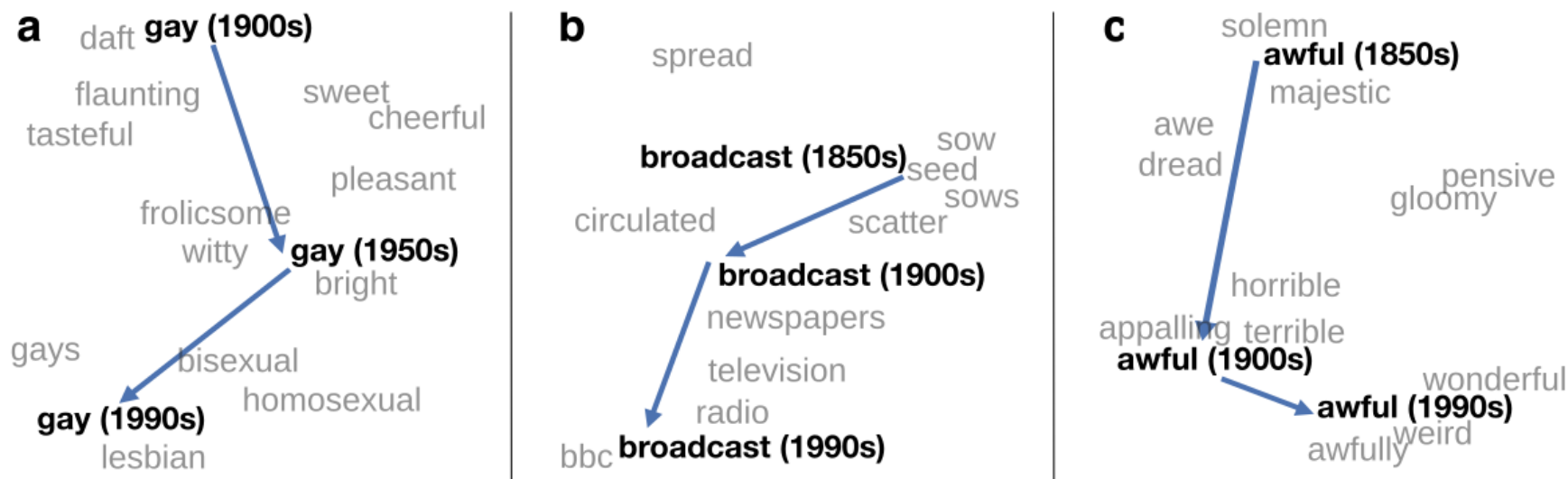


Words in different languages but with similar meanings (i.e. translations) are represented by similar vectors

Used in Machine Translation

# Word History through Embeddings

Train embeddings on old books to study changes in word meaning



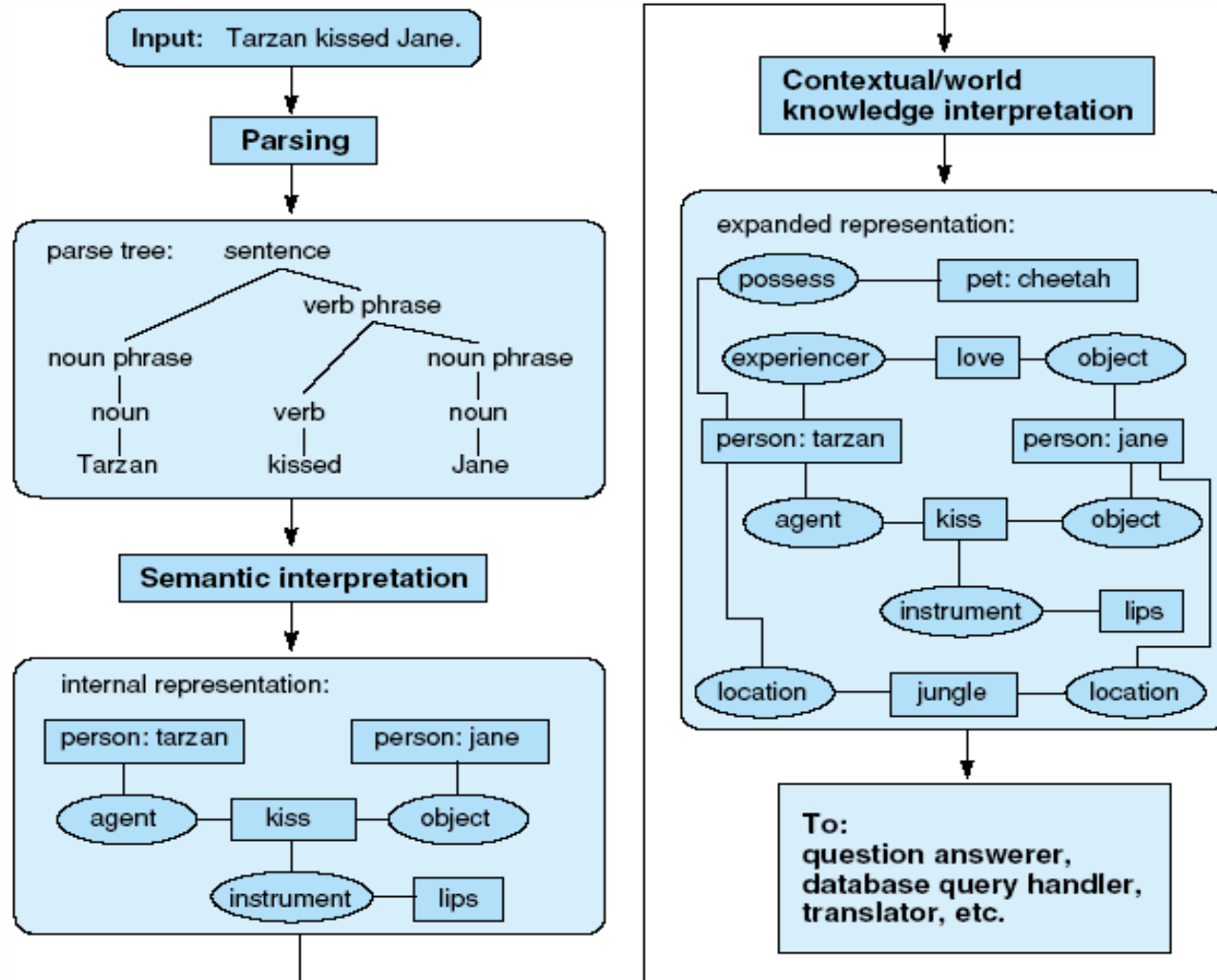
Project 300 dimensions down into 2

# Today

1. Motivation
2. Word Embeddings
3. Deep Learning for NLP



# Stages of NLU



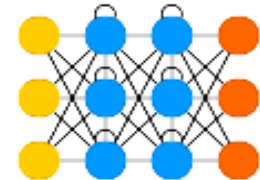
# Deep Learning for NLP

Deep learning models for NLP use:

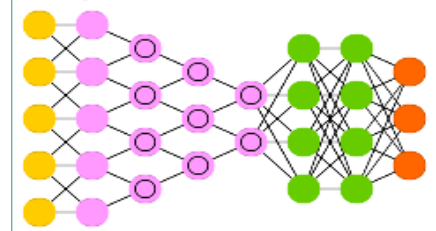
- Vector representation of words
  - i.e., word embeddings
- Neural network structures
  - Recurrent Neural Networks (RNNs)
  - Recursive Neural Networks
  - Convolutional Networks (CNNs)
  - ...



Recurrent Neural Network (RNN)

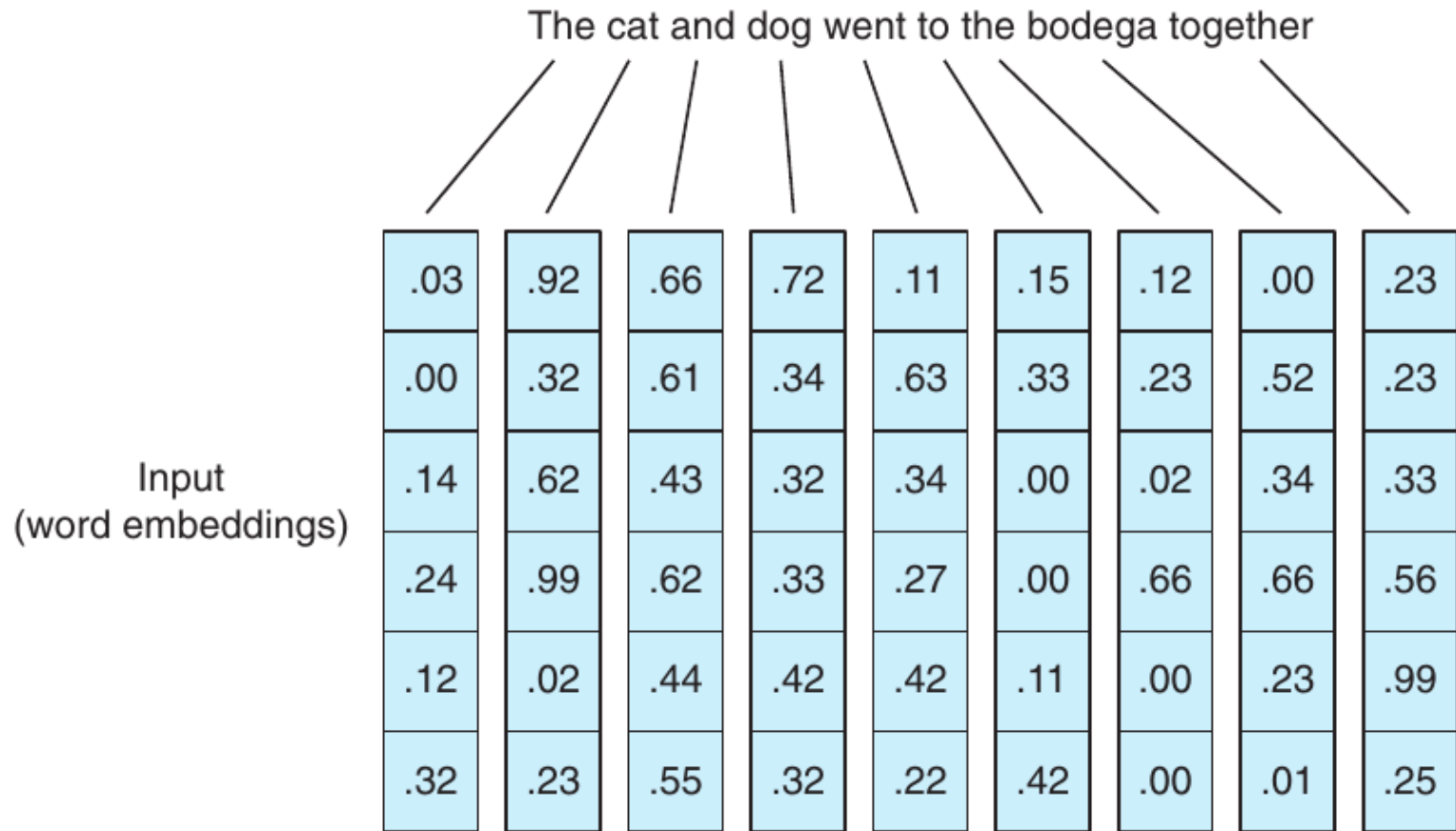


Deep Convolutional Network (DCN)

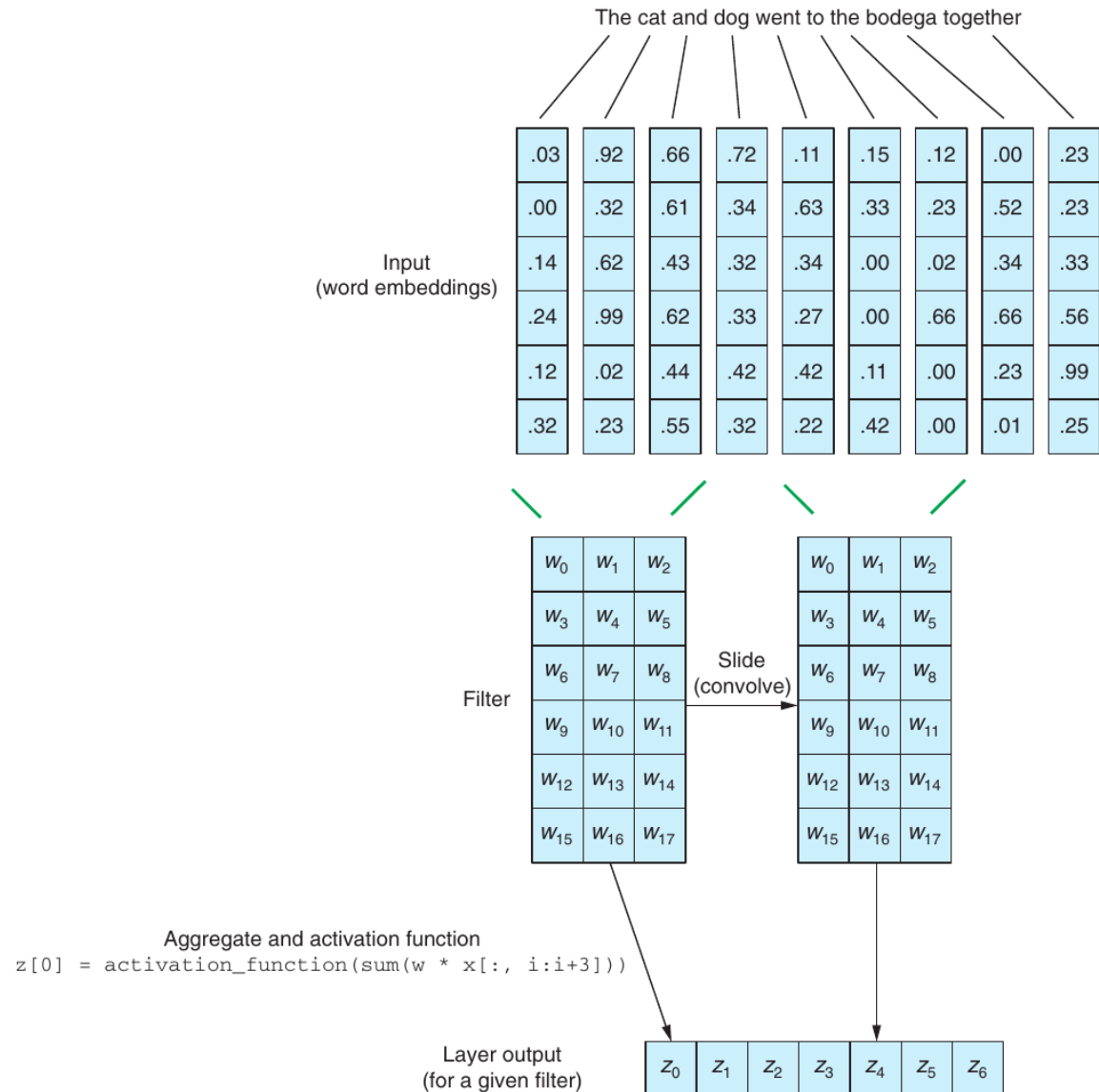




# CNNs for NLP

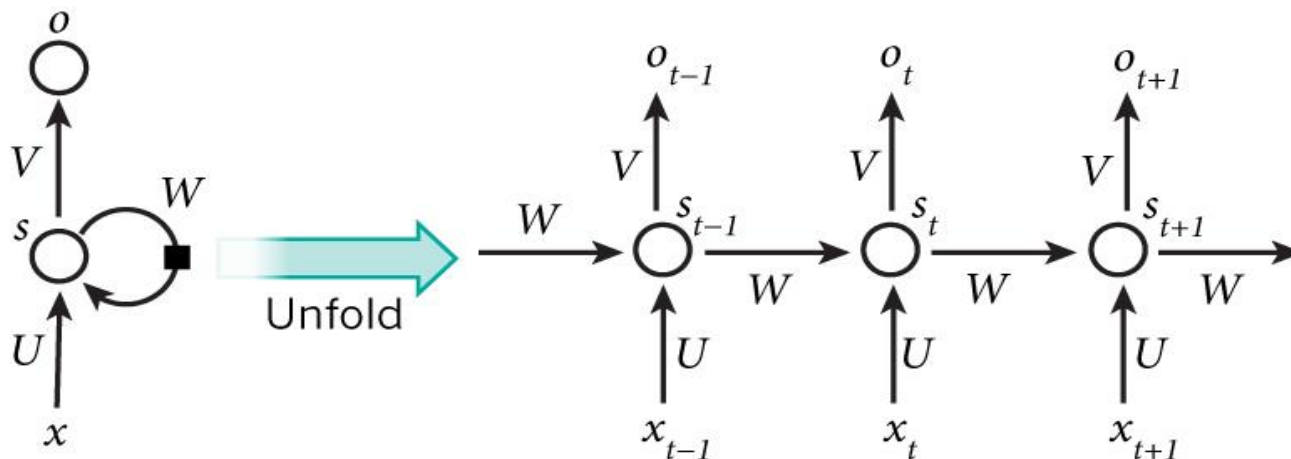


# CNNs for NLP (II)



# Recurrent Neural Networks

- To model sequences of decisions, such as machine translation, language modelling, ...
  - e.g., A word at position  $n$  can influence a word/decision at position  $n+t$
- decision/output from the past can influence current decision/output
- Networks with loops in them, allowing information to persist.



# Cool Applications

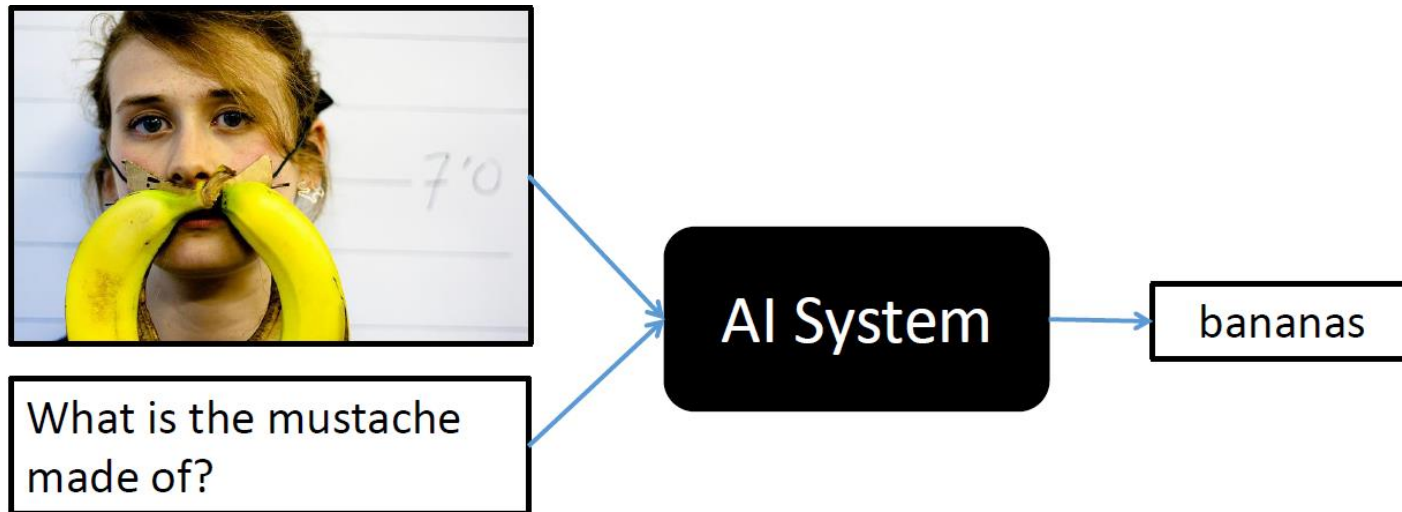
- RNNs + word embeddings
  - Google Translate
    - As of July 2017, uses an RNN + word embeddings (called Neural Machine Translation (NMT) )
    - Input sequence: words in a source sentence
    - Output sequence: words in the target language
  - Dialogue Systems
- CNNs + RNNs + word embeddings
  - Image Captioning
  - Video to Text Descriptions
  - Visual Question Answering
- ...

# Visual Question Answering

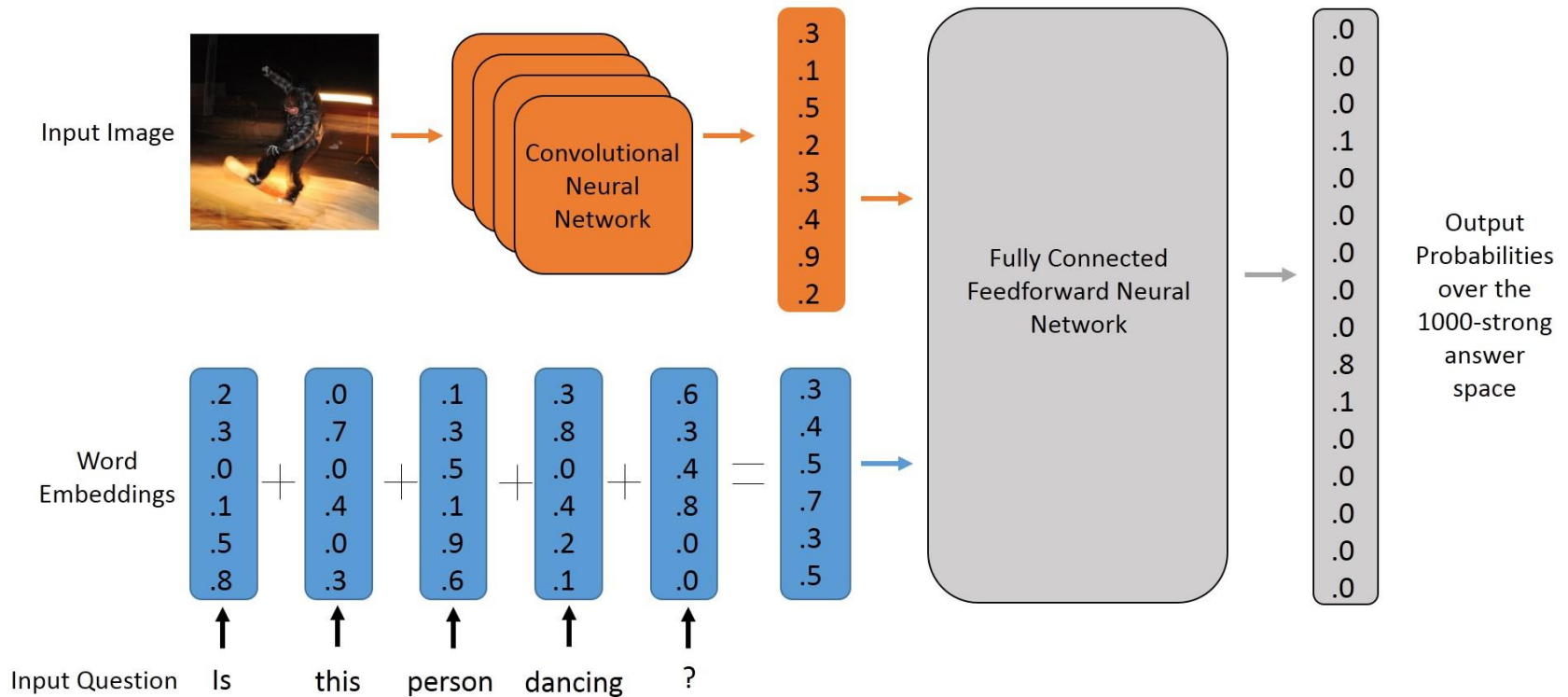
## Demo Website

Input: an image + a natural language question

Output: natural language answer

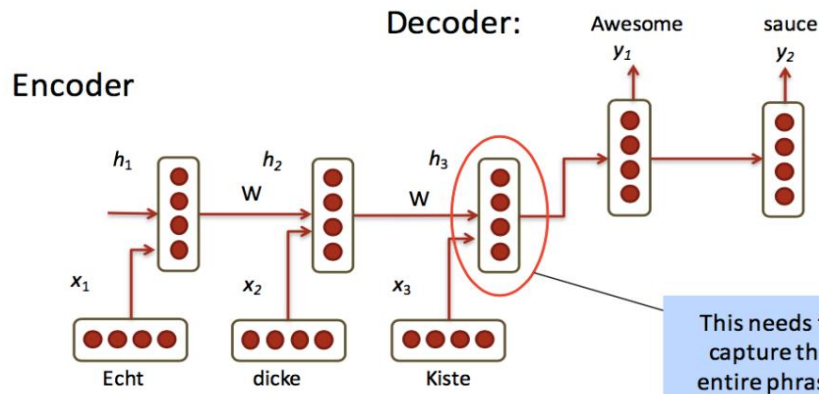


# Visual Question Answering

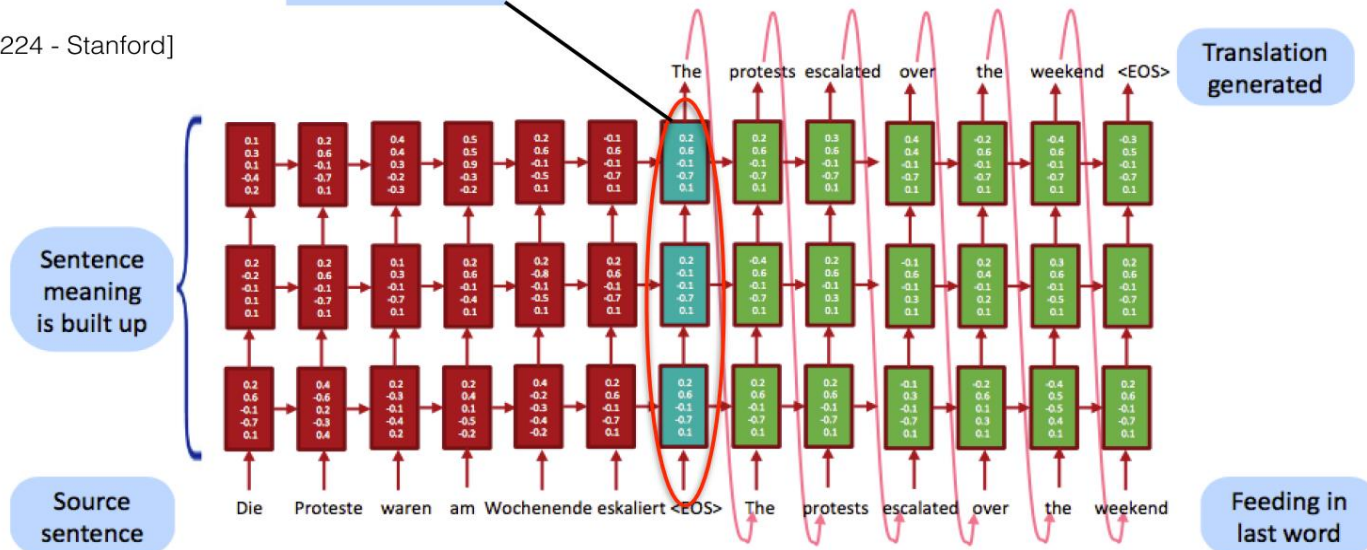


- The output is conditioned on both image and textual inputs.
- A CNN is used to encode the image.
- A RNN is used to encode the sentence.

# Machine Translation

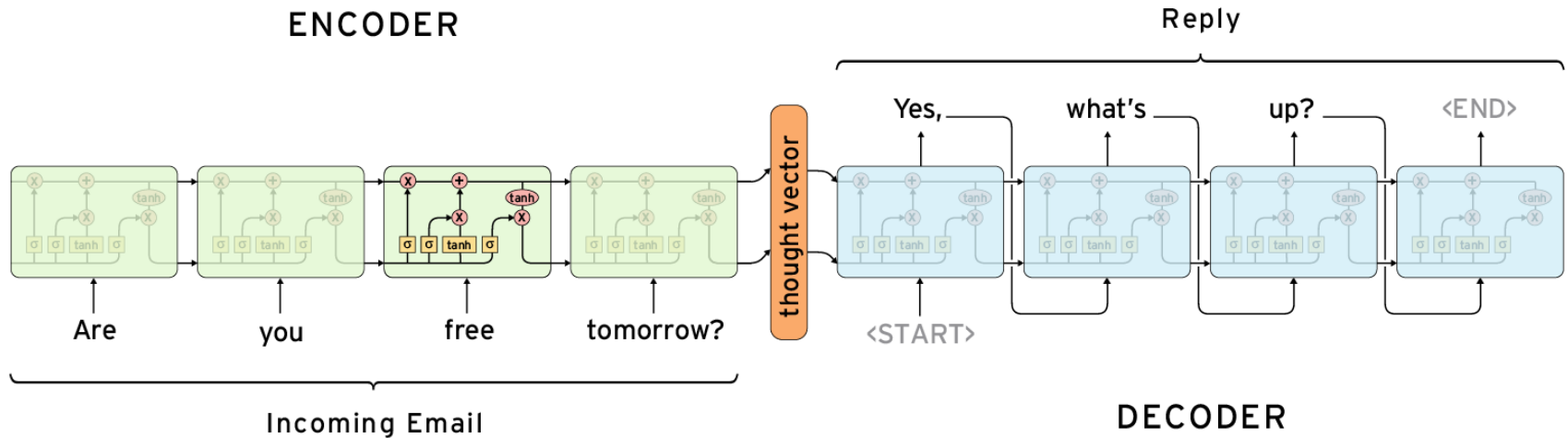


[CS224 - Stanford]



[CS224 - Stanford]

# Conversational Agents



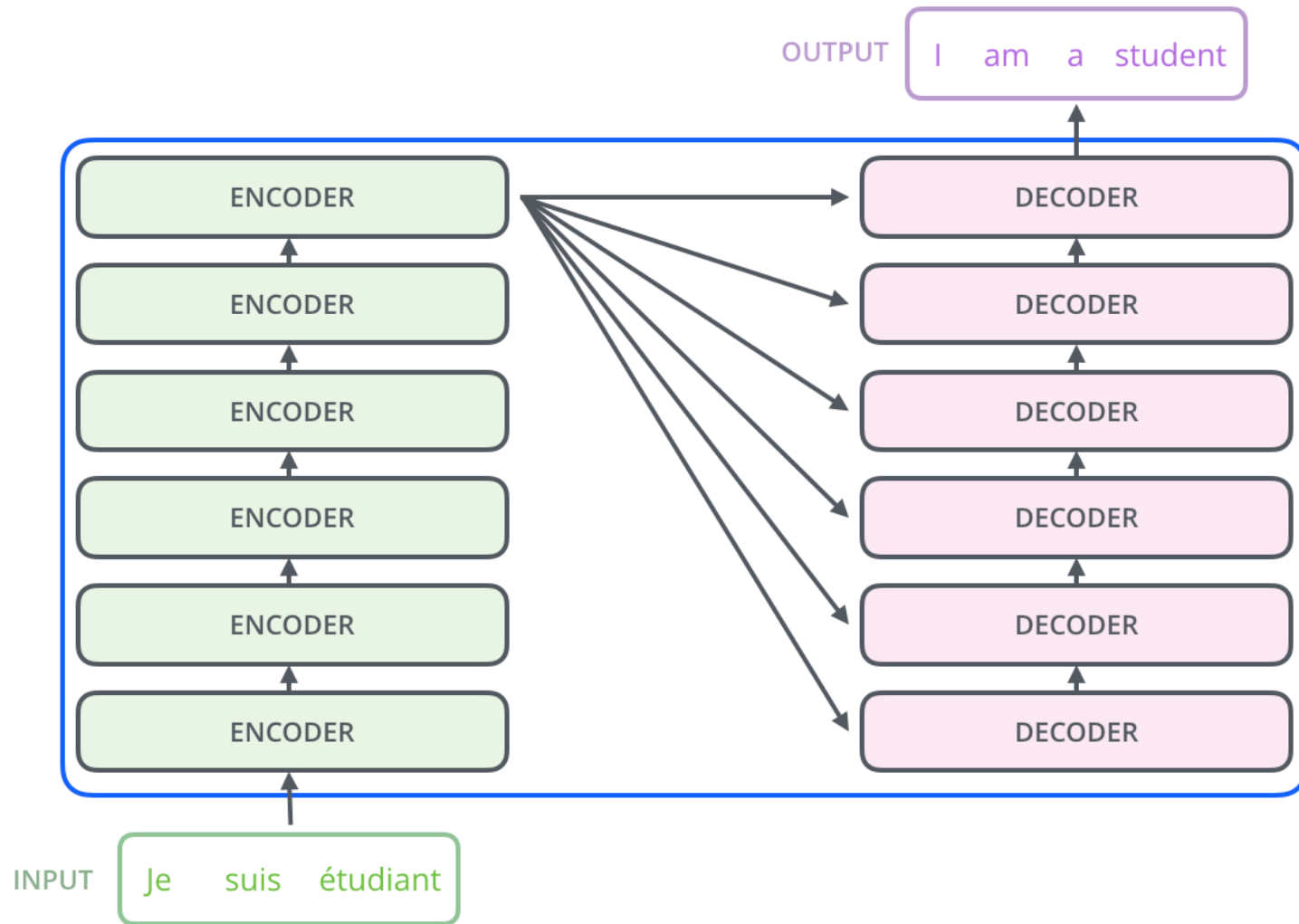


# Transformers

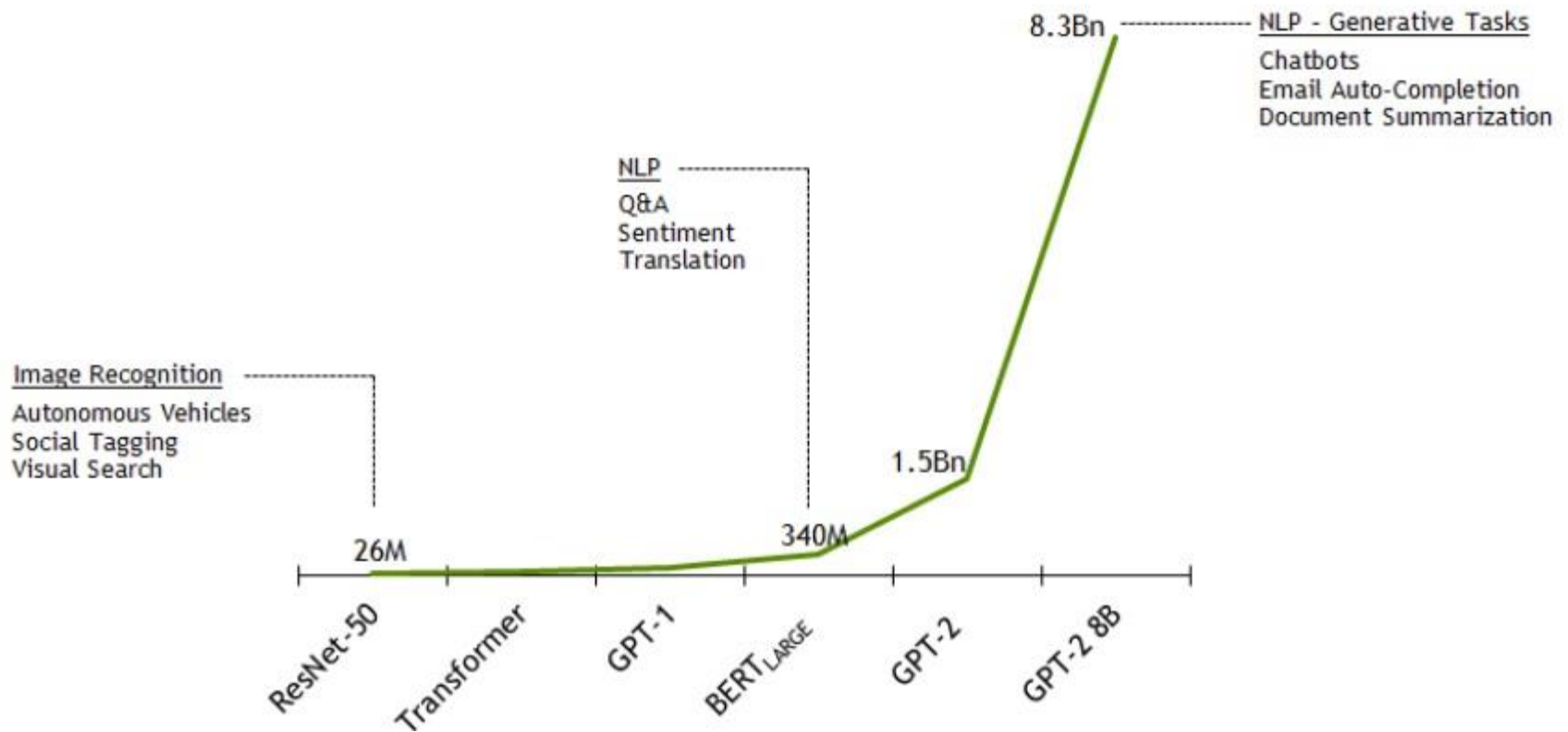


<http://jalammar.github.io/illustrated-transformer/>

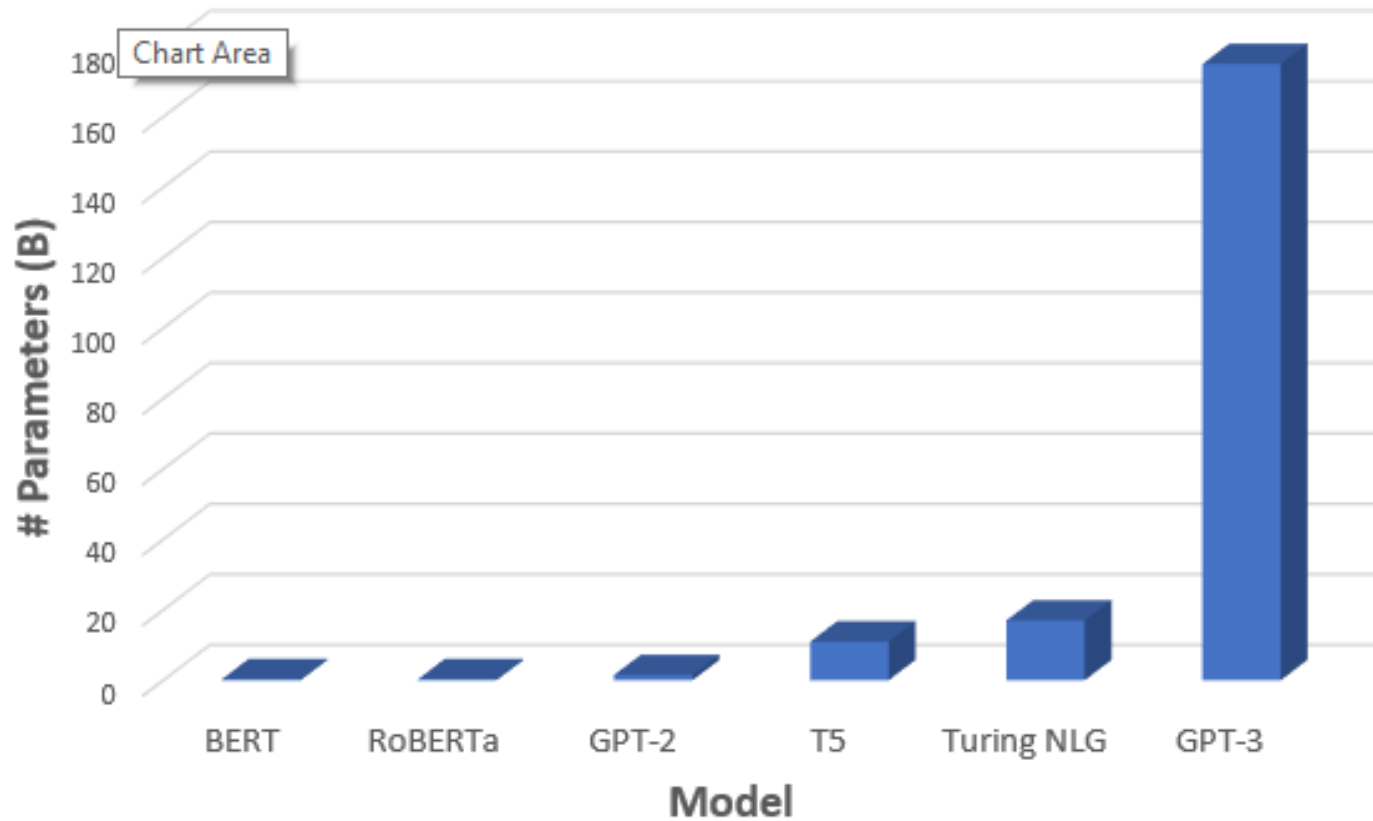
# Transformers (II)



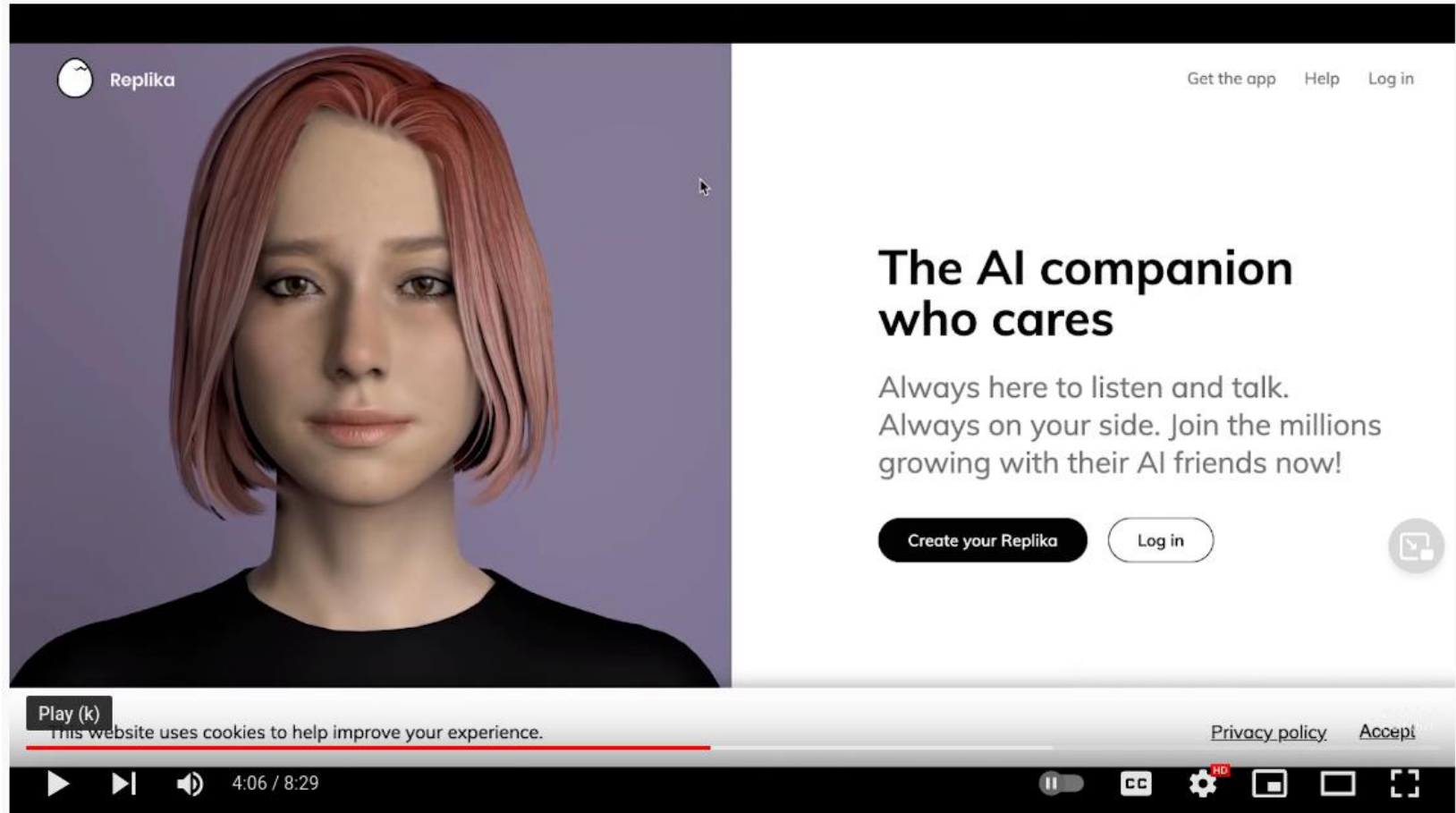
# Transformers (II)



# GPT-3



# GPT-3 Applications



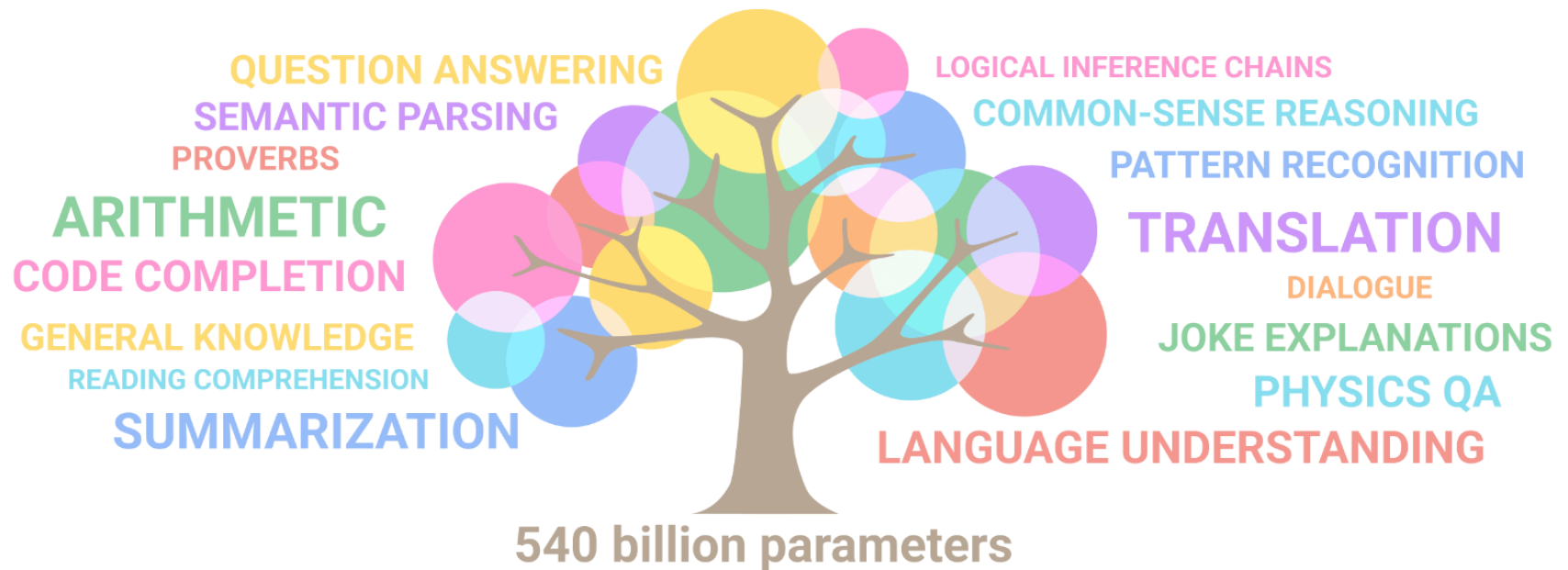
TORONTO

9 Cool Startups Powered by GPT-3

<https://www.youtube.com/watch?v=QYfZJ5jbUWU>

# Google PaLM (April 2022)

Pathways Language Model (PaLM):  
Scaling to 540 Billion Parameters for Breakthrough Performance



# Google PaLM: Explaining Jokes!

I will explain these jokes:

(1) The problem with kleptomaniacs is that they always take things literally.

Explanation: This joke is wordplay. Someone who "takes things literally" is someone who doesn't fully understand social cues and context, which is a negative trait. But the definition of kleptomania is someone who literally takes things.

(2) Always borrow money from a pessimist. They'll never expect it back.

Explanation: Most people expect you to pay them back when you borrow money, however a pessimist is someone who always assumes the worst, so if you borrow money from them, they will expect that you won't pay them back anyways.

**Input:** I was supposed to start writing the paper at 5:00 PM. But then I started playing with this cool new language model for 10 minutes. 10 minutes later, it's suddenly 9:30 PM!

**Model Output:** This joke is about how time flies when you're having fun. The person was supposed to start writing a paper at 5:00 PM, but they started playing with a language model instead. They thought they were only playing with it for 10 minutes, but it was actually 4.5 hours.

**Input:** I was going to fly to visit my family on April 6th. My mom said "Oh great, your step dad's poetry reading is that night!" So now I'm flying in on April 7th.

**Model Output:** The joke is that the speaker's mother is trying to get them to go to their step dad's poetry reading, but the speaker doesn't want to go, so they are changing their flight to the day after the poetry reading.

**Input:** Reading "War and Peace" doesn't scare me. I once had to review the CL for my intern's whole project on his final day!

**Model Output:** 'War and Peace' is a very long book, so reading it is a daunting task. However, the speaker is saying they had to review the code for their intern's project on their final day, which is even more daunting.

**Input:** I think I understand why me and my partner are such a good match. I get excited when I smell food and become super alert when the doorbell rings. My girlfriend Shelley becomes ecstatic when she sees a person who she just saw 5 hours ago.

**Model Output:** This joke is about dogs. Dogs get excited when they smell food and when the doorbell rings. The girlfriend is excited to see someone she just saw 5 hours ago, which is a trait of dogs.

**Input:** I tried 10,000 random restarts of my neural network, but I was accused of overfitting. I guess no good seed goes unpunished.

**Model Output:** This joke is a pun. A neural network is a computer program that can learn from data. A "seed" is a number that is used to initialize a random number generator. A "good seed" is a number that produces a good random number generator. The phrase "no good deed goes unpunished" means that if you do something good, you will be punished for it.

**Input:** What's the difference between a zebra and an umbrella? One is a striped animal related to horses, another is a device you use to stop the rain from falling on you.

**Model Output:** This joke is an anti-joke. The joke is that the answer is obvious, and the joke is that you were expecting a funny answer.

# Today

1. Motivation
2. Word Embeddings
3. Deep Learning for NLP