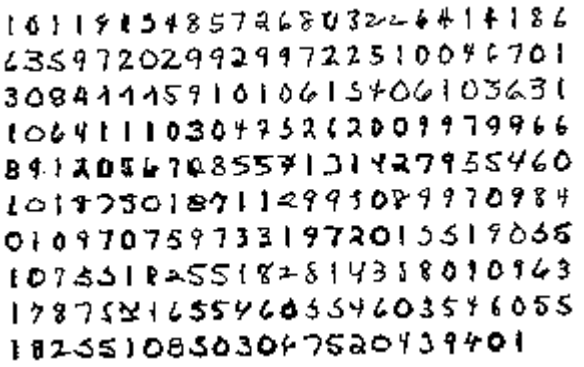
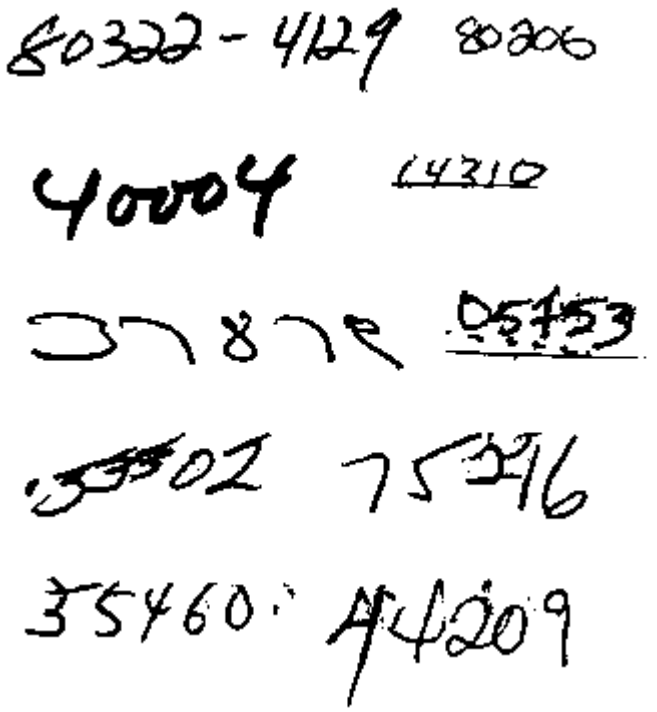
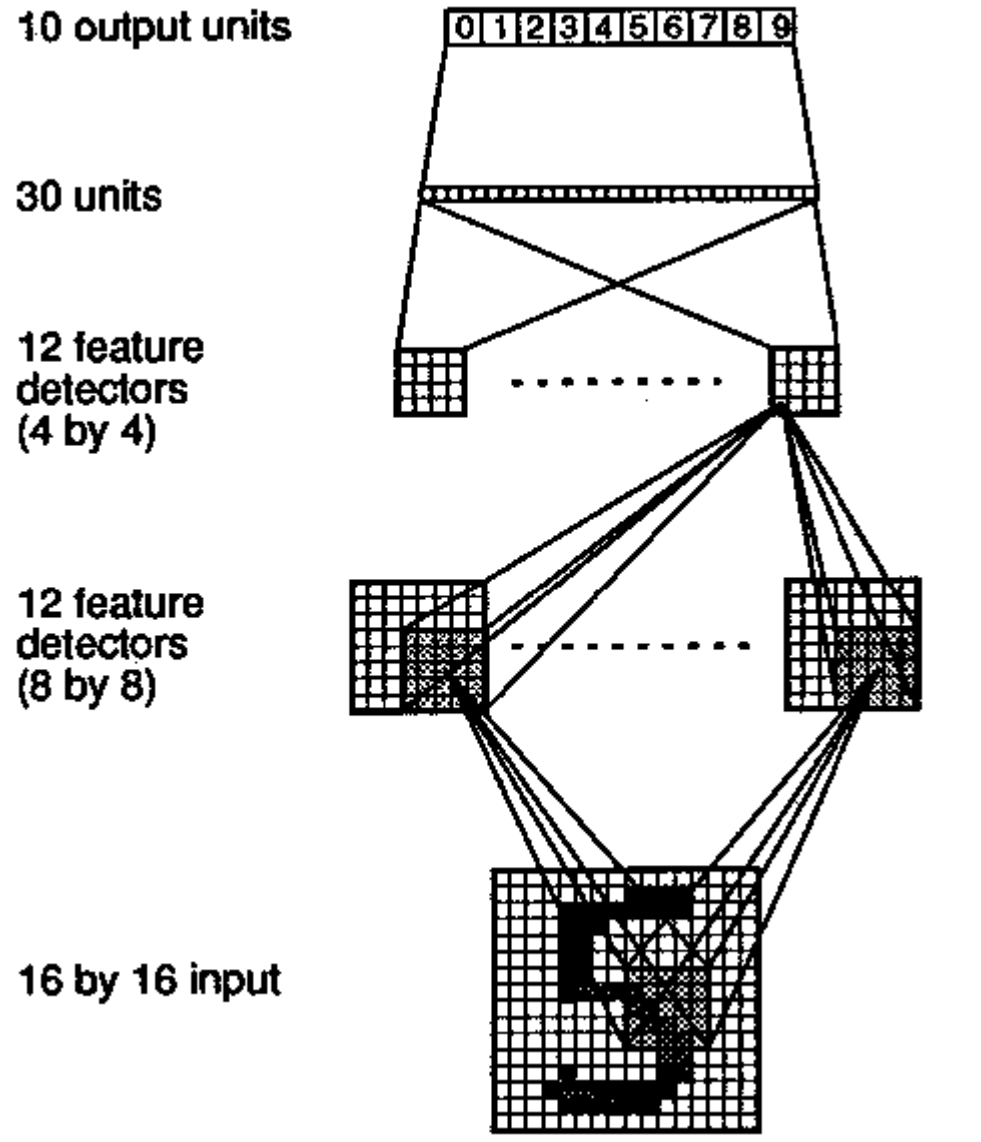
Real-World Neural Network Applications

- **Zip code recognizer** (Yann LeCun, AT&T Bell Labs, 1980s)
  - Used a large neural network with several layers
  - Trained on handwritten zip codes from U.S. mail
  - Achieved the state of the art in digit recognition
  - Classification accuracy > 95%



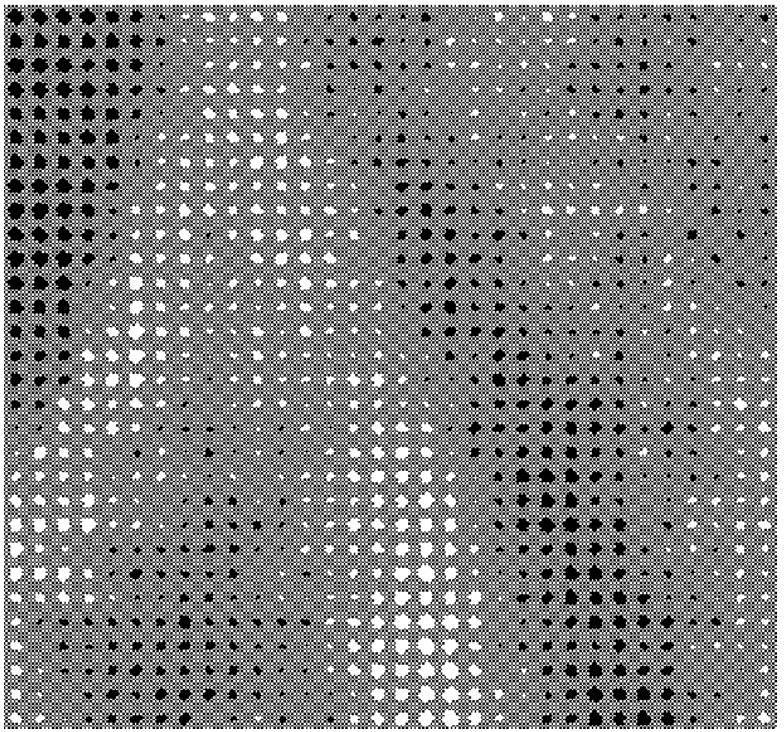


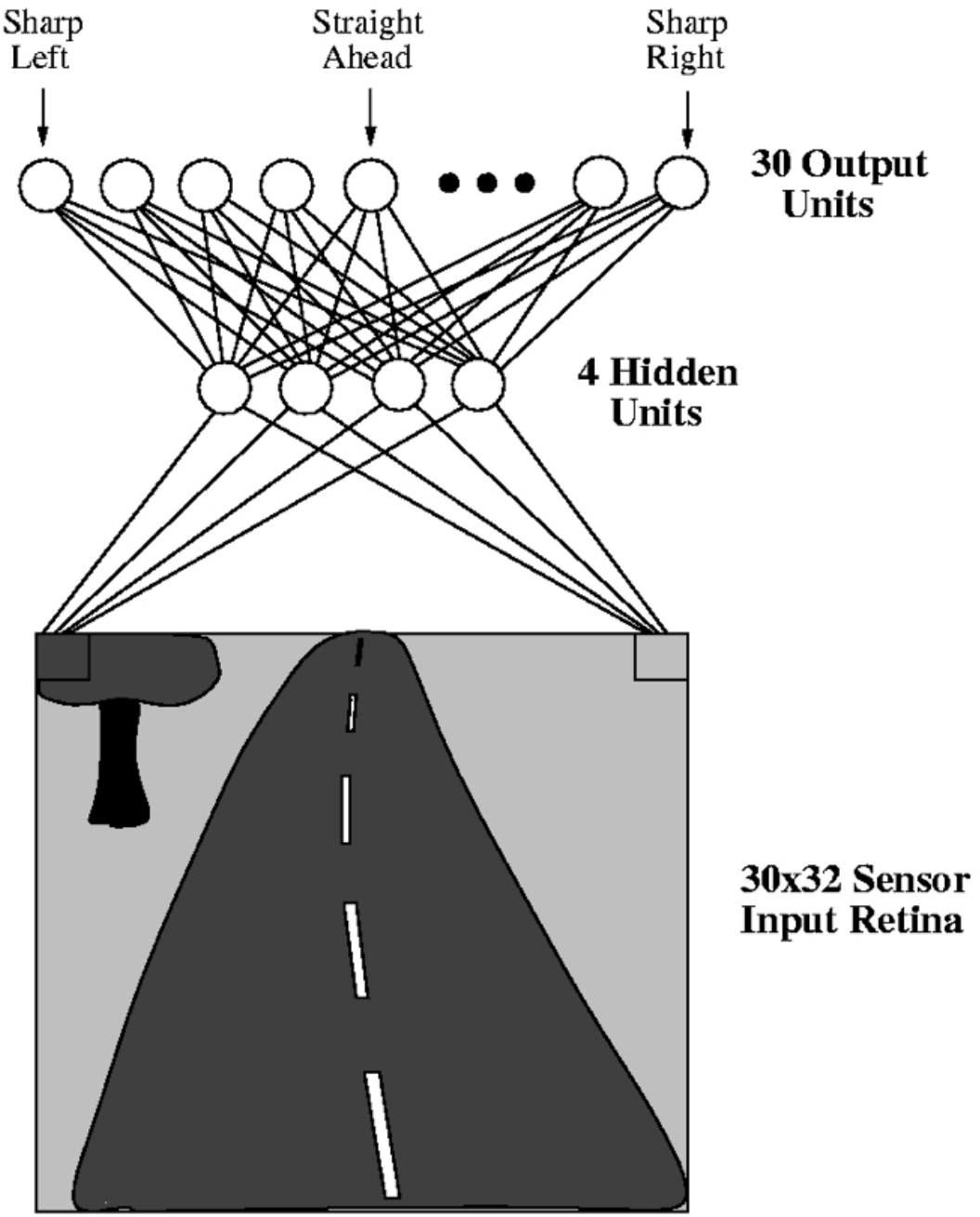
Real-World Neural Network Applications



Real-World Neural Network Applications

- **ALVINN** (Dean Pomerleau, CMU, 1990s)
  - Autonomous vehicle controlled by a neural network
  - Input: image of road, Output: steering wheel position
  - Neural network learns by “observing” a human driver
  - In 1995, steered a car semi-autonomously from coast to coast (all but 50 of 2,850 miles)



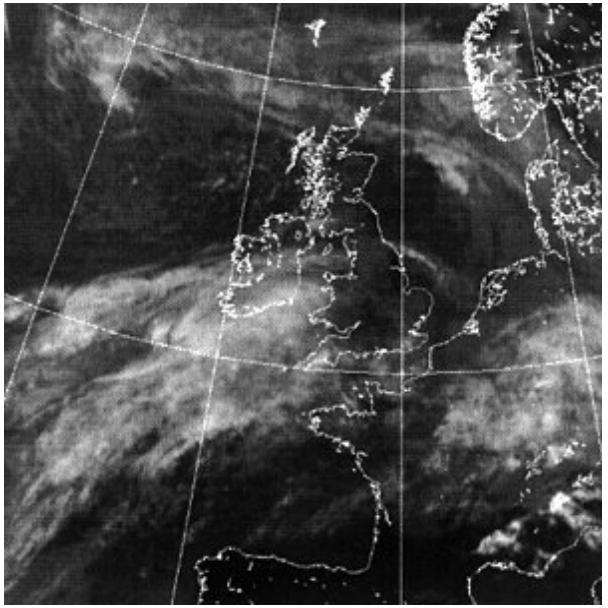

Sharp
Left
Straight
Ahead
Sharp
Right
30 Output
Units
4 Hidden
Units
30x32 Sensor
Input Retina

Real-World Neural Network Applications

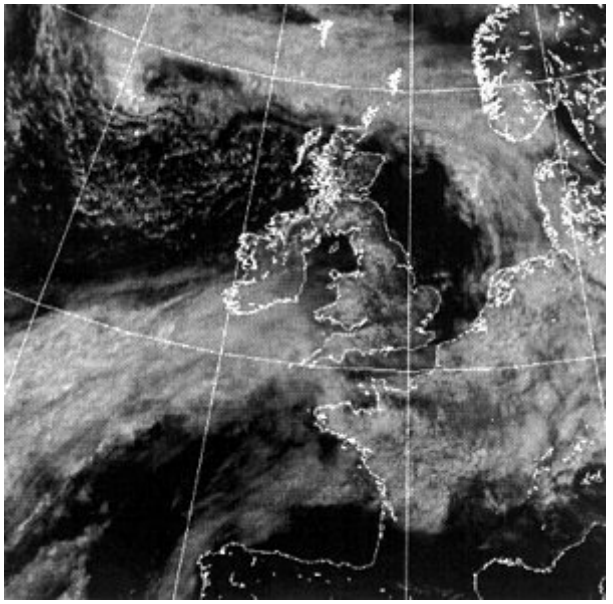
- **Weather forecasting** (UK Meteorological Office)
  - Visible and infrared satellite images were used to train a neural network to classify cloud patterns into 4 categories:
    - clear land/sea
    - dynamic cloud
    - shallow convective cloud
    - deep convective cloud
  - During training, statistical features were calculated for each image and presented to the network, along with the classification category
  - Network achieved a prediction accuracy of 94% on independent image samples
  - System became fully operational in 1999

Real-World Neural Network Applications

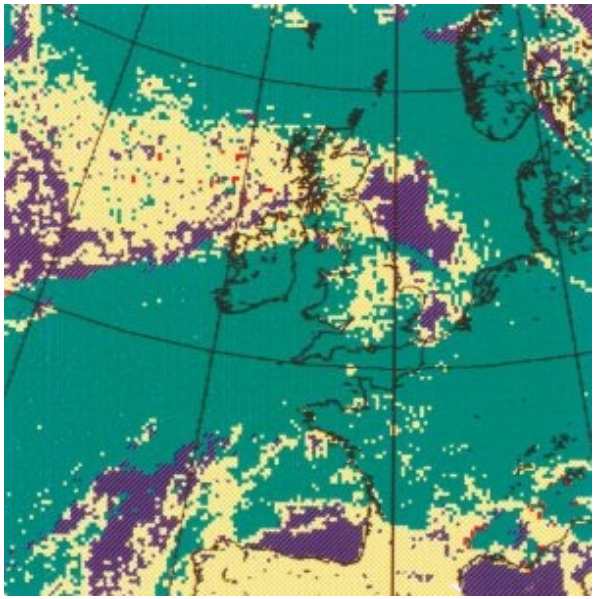
Visible



Infrared



⇒

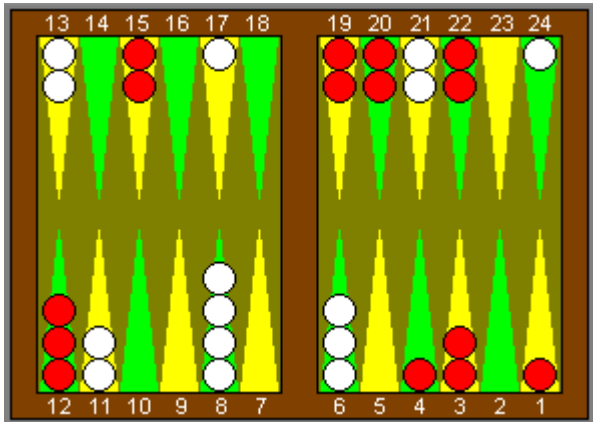


Expected airmass

purple = clear land/sea
green = dynamic cloud
yellow = shallow convective cloud
red = deep convective cloud

Real-World Neural Network Applications

- **TD-Gammon** (Gerry Tesauro, IBM, 1990s)
  - Learned by playing over 1.5 million games against itself
  - Discovered novel board evaluation strategies
  - Trained neural networks with reinforcement learning
  - Achieved parity with the top 5-10 players in the world
  - By far the best computer backgammon program

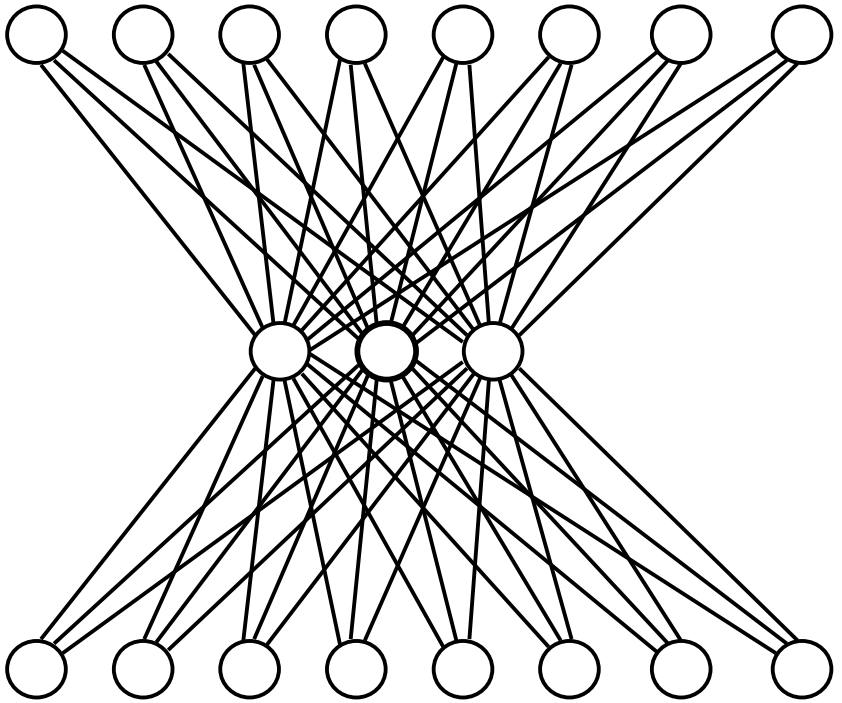


Auto-Associator Networks

8 output units

3 hidden units

8 input units



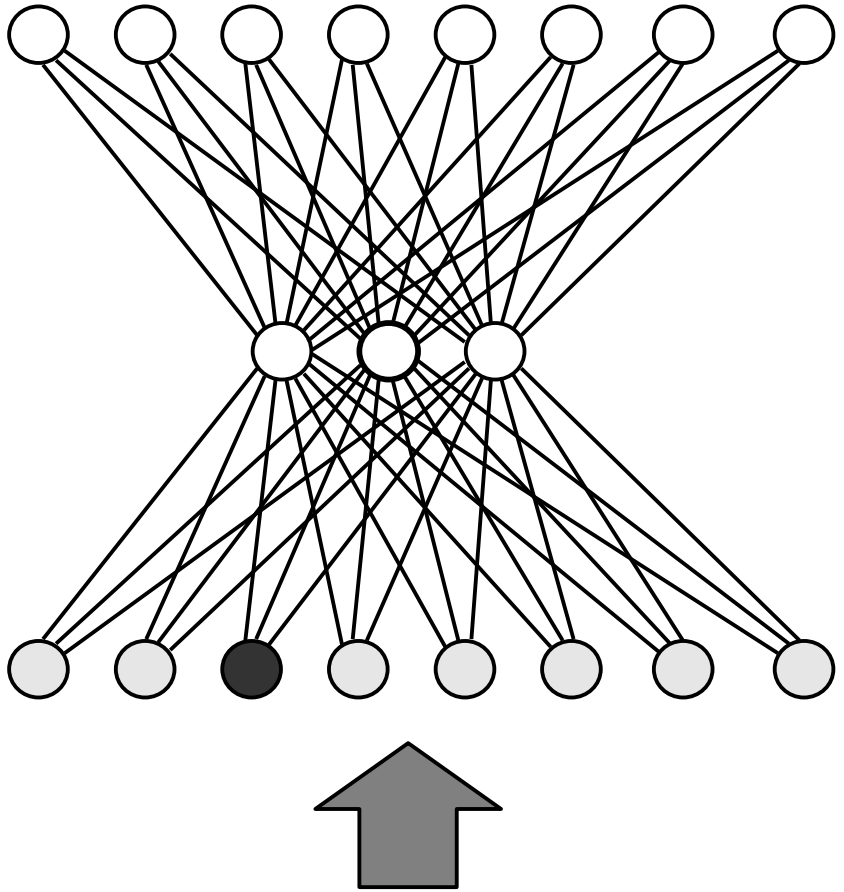
Auto-Associator Networks

Target = [0 0 1 0 0 0 0 0]

8 output units

3 hidden units

8 input units



Input = [0 0 1 0 0 0 0 0]

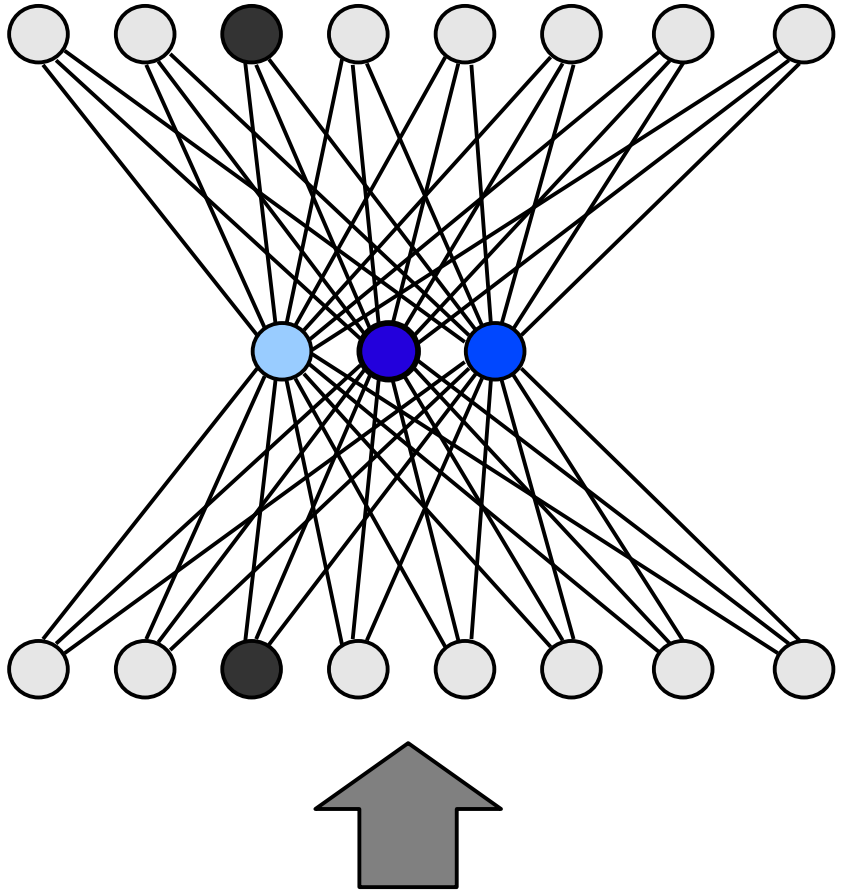
Auto-Associator Networks

Target = [0 0 1 0 0 0 0 0]

8 output units

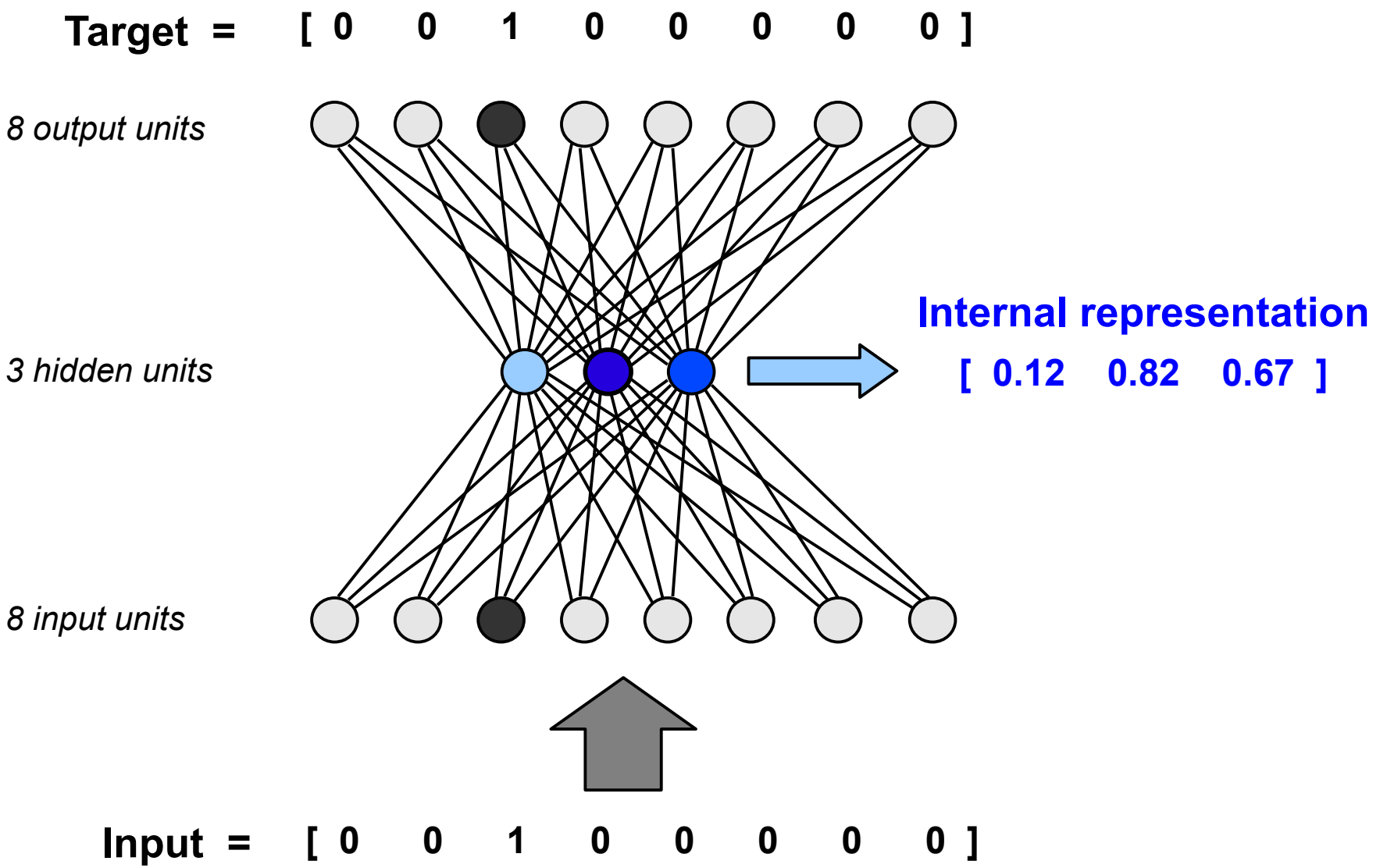
3 hidden units

8 input units



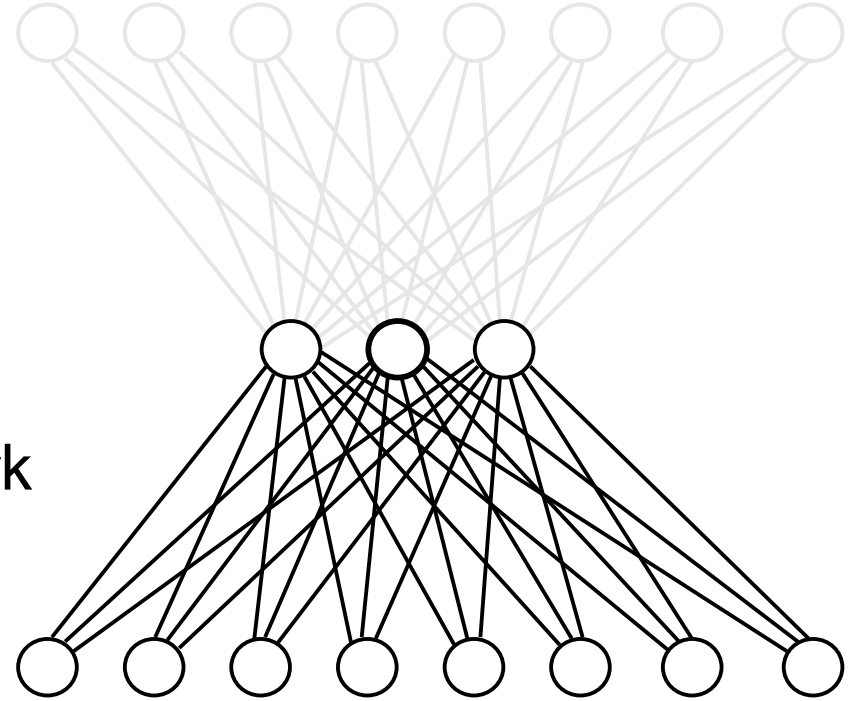
Input = [0 0 1 0 0 0 0 0]

Auto-Associator Networks



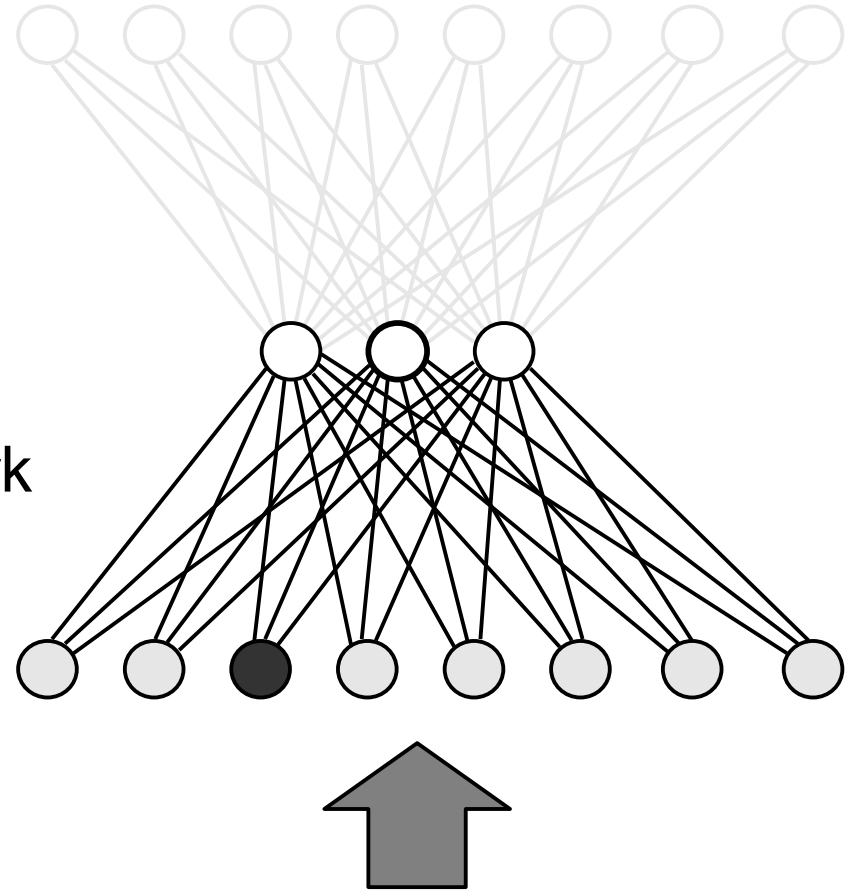
Auto-Associator Networks

Encoder network



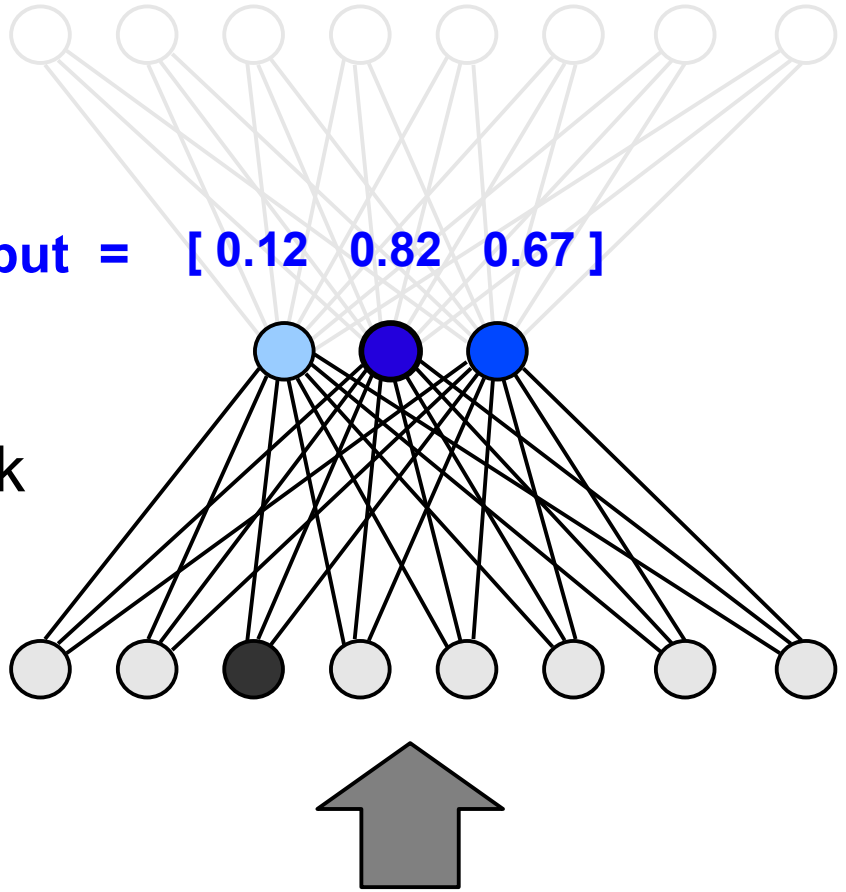
Auto-Associator Networks

Encoder network



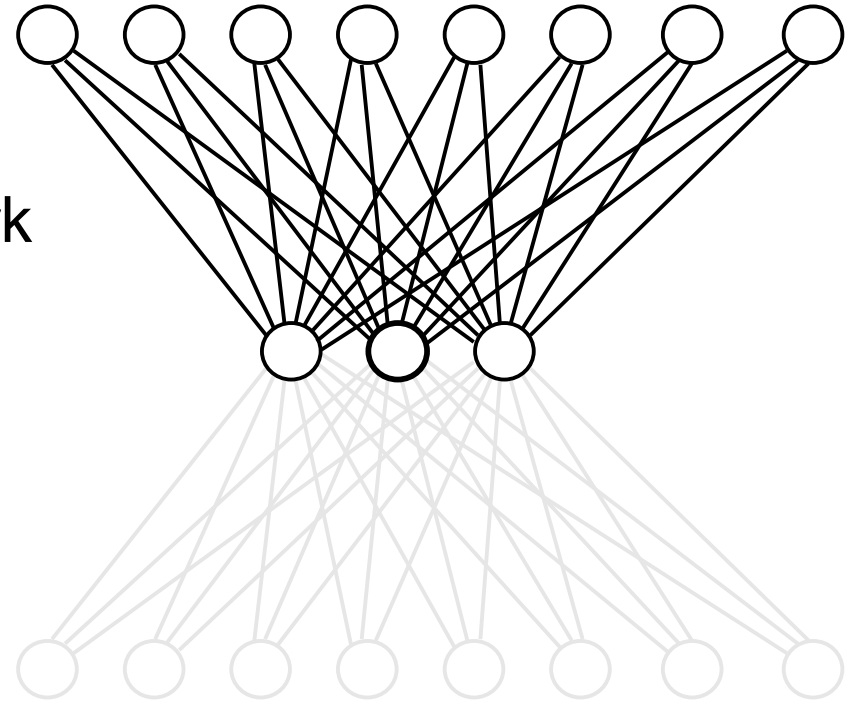
Input = **[0 0 1 0 0 0 0 0]**

Auto-Associator Networks



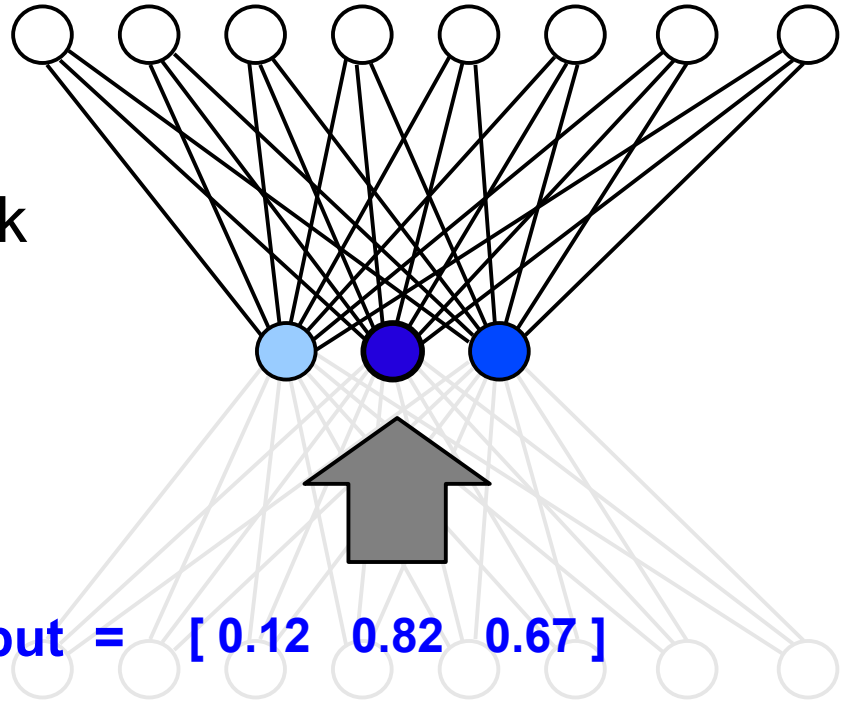
Auto-Associator Networks

Decoder network



Auto-Associator Networks

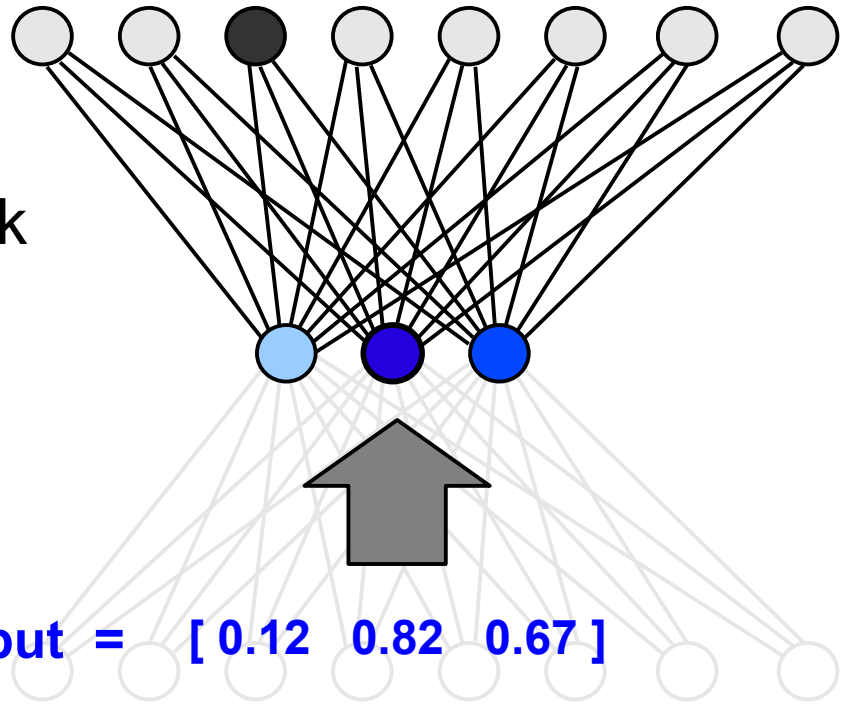
Decoder network



Input = [0.12 0.82 0.67]

Auto-Associator Networks

Output = [0 0 1 0 0 0 0 0]



Decoder network

Input = [0.12 0.82 0.67]

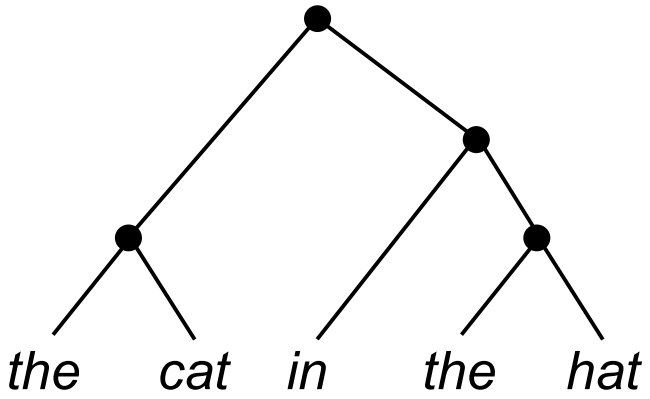
Auto-Associator Demo

Using RAAM for Linguistic Processing

- Jerry Fodor and Zenon Pylyshyn published a strongly negative critique of connectionism in 1988
- *Connectionism and Cognitive Architecture: A Critical Analysis* (Chapter 12 of *Mind Design II*)
- They claimed that connectionist networks are incapable of representing or processing linguistic structure
- Jordan Pollack developed the **RAAM model** to show how networks could encode hierarchical symbol structures
- David Chalmers showed how networks could process RAAM-encoded hierarchical structures **holistically**
- His network transformed sentences from active to passive
- See Chapter 7: “The Second AI Debate” in *Artificial Minds*

RAAM: Recursive Auto-Associative Memory

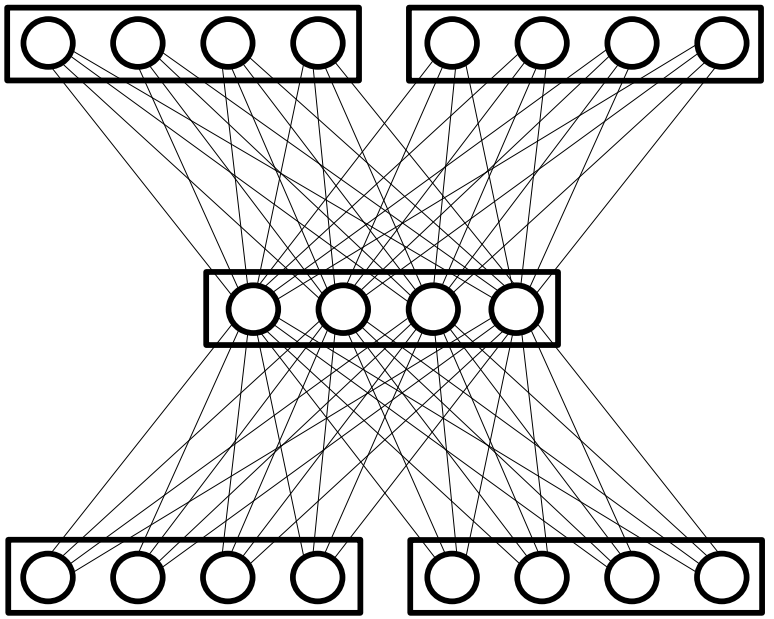
- Developed by Jordan Pollack
- A method of encoding hierarchically structured symbolic information suitable for processing by a neural network
- How to represent the noun phrase "*the cat in the hat*" ?
- As a list of symbols: **`((the cat) (in (the hat)))`**
- As a parse tree:



- How to encode this as a distributed numerical representation?

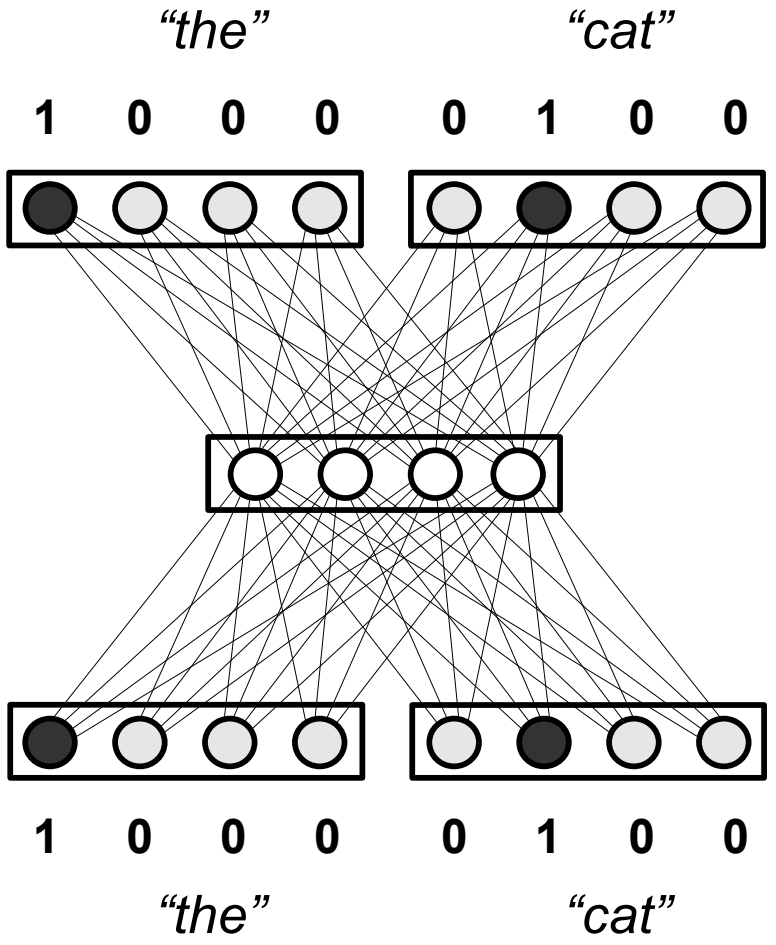
RAAM: Recursive Auto-Associative Memory

“the” = **1 0 0 0**
“cat” = **0 1 0 0**
“in” = **0 0 1 0**
“hat” = **0 0 0 1**



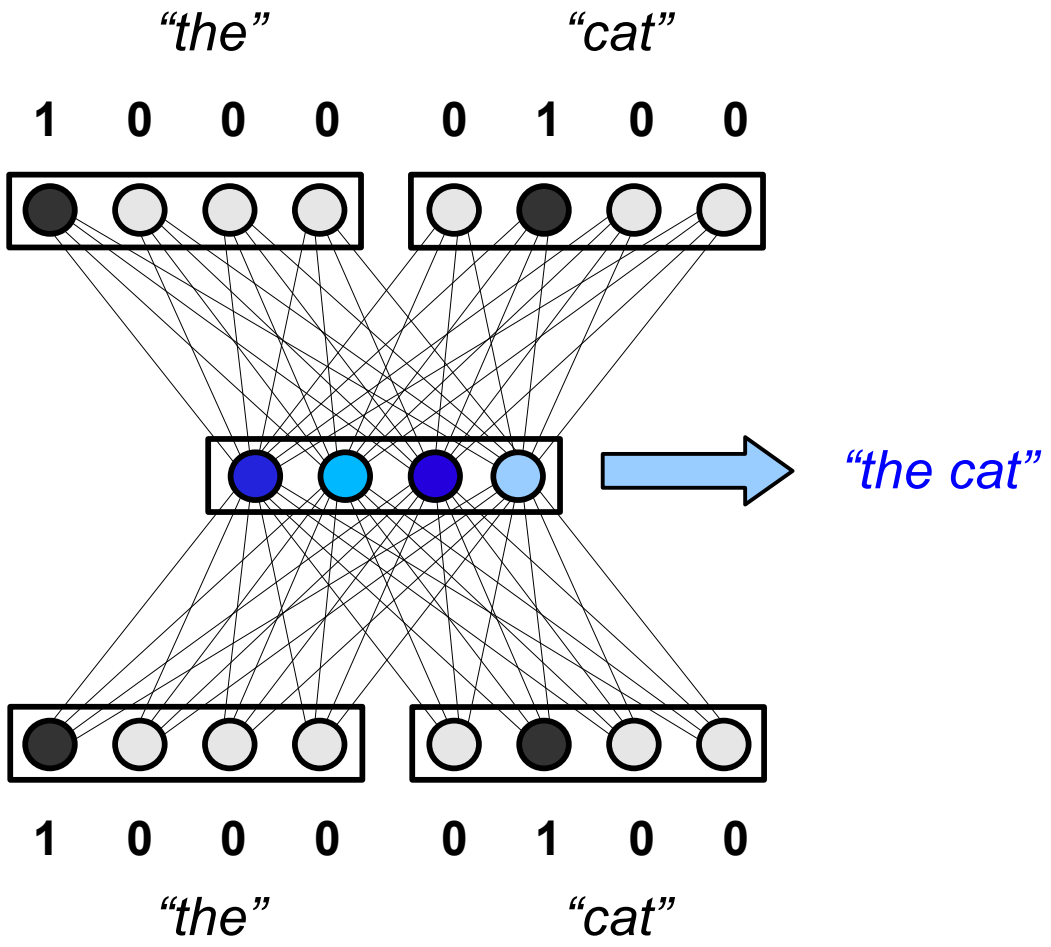
RAAM: Recursive Auto-Associative Memory

"the" = **1 0 0 0**
"cat" = **0 1 0 0**
"in" = **0 0 1 0**
"hat" = **0 0 0 1**



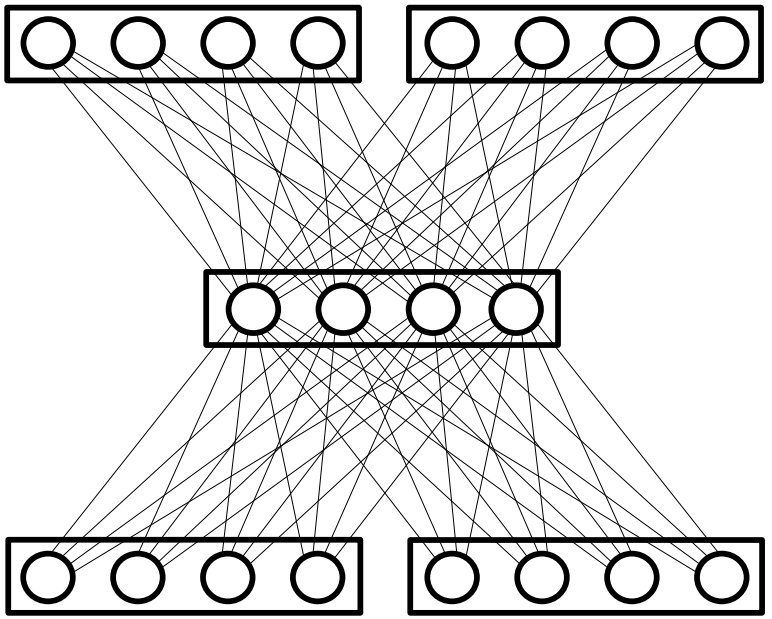
RAAM: Recursive Auto-Associative Memory

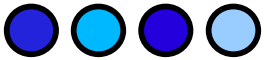
"the" = **1 0 0 0**
"cat" = **0 1 0 0**
"in" = **0 0 1 0**
"hat" = **0 0 0 1**



RAAM: Recursive Auto-Associative Memory

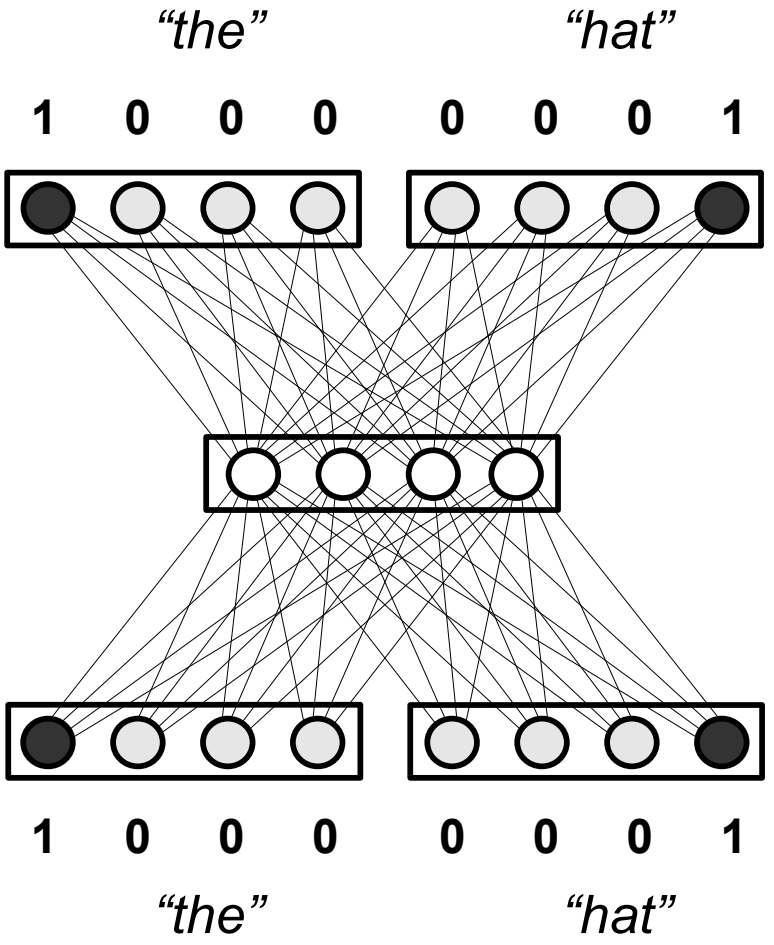
“the” = **1 0 0 0**
“cat” = **0 1 0 0**
“in” = **0 0 1 0**
“hat” = **0 0 0 1**



“the cat” = 

RAAM: Recursive Auto-Associative Memory

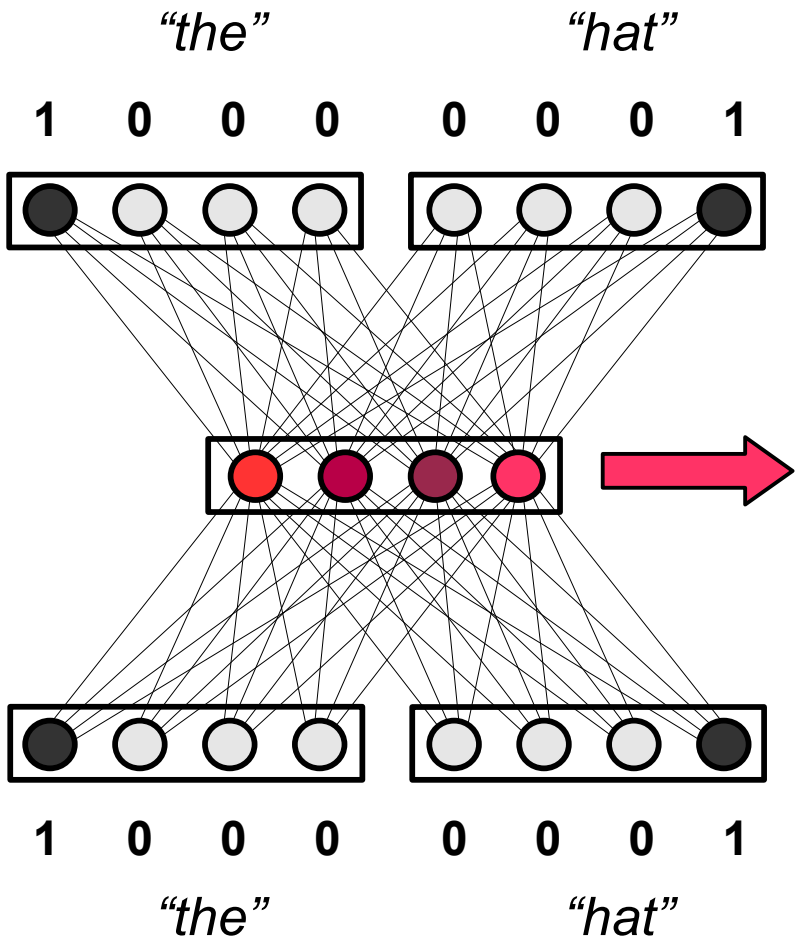
"the" = **1 0 0 0**
"cat" = **0 1 0 0**
"in" = **0 0 1 0**
"hat" = **0 0 0 1**



"the cat" =

RAAM: Recursive Auto-Associative Memory

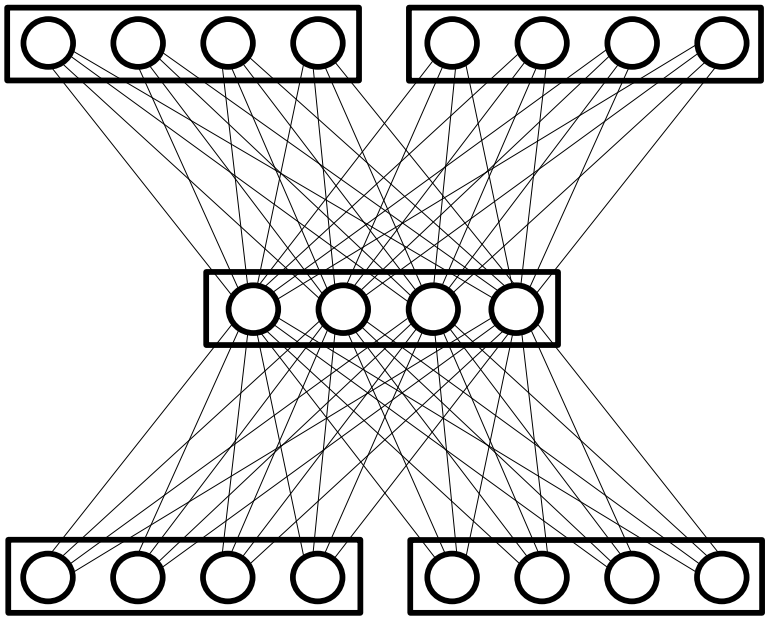
"the" = **1 0 0 0**
"cat" = **0 1 0 0**
"in" = **0 0 1 0**
"hat" = **0 0 0 1**



"the cat" =

RAAM: Recursive Auto-Associative Memory

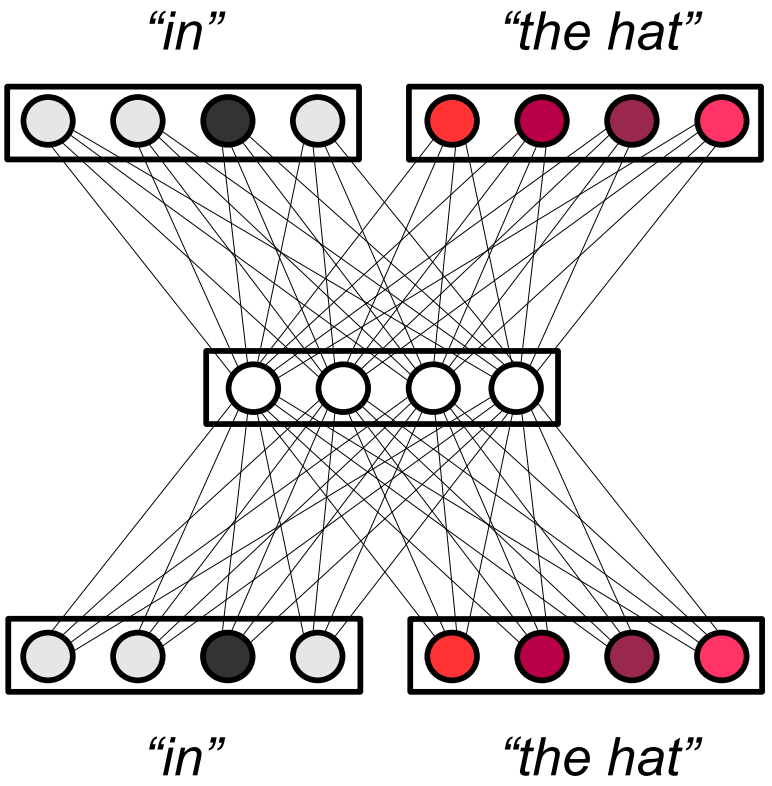
“the” = **1 0 0 0**
“cat” = **0 1 0 0**
“in” = **0 0 1 0**
“hat” = **0 0 0 1**



“the cat” =
“the hat” =

RAAM: Recursive Auto-Associative Memory

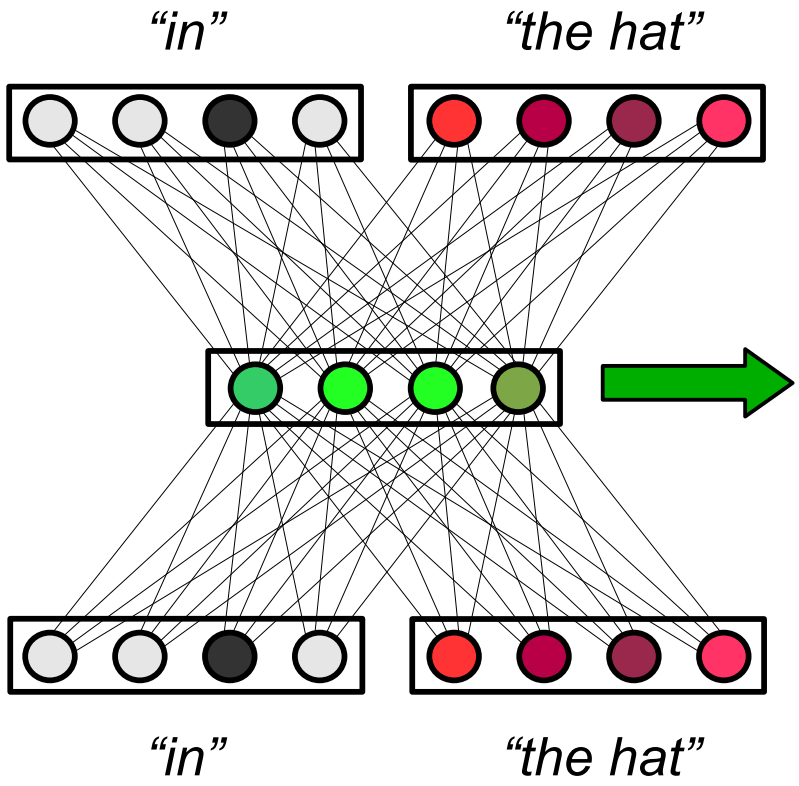
"the" = **1 0 0 0**
"cat" = **0 1 0 0**
"in" = **0 0 1 0**
"hat" = **0 0 0 1**



"the cat" =
"the hat" =

RAAM: Recursive Auto-Associative Memory

"the" = **1 0 0 0**
"cat" = **0 1 0 0**
"in" = **0 0 1 0**
"hat" = **0 0 0 1**

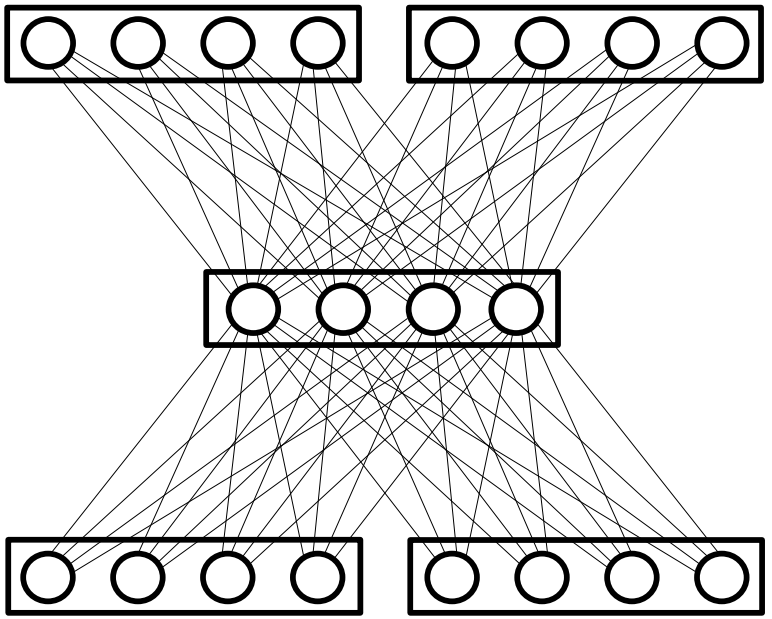


"in the hat"

"the cat" =
"the hat" =

RAAM: Recursive Auto-Associative Memory

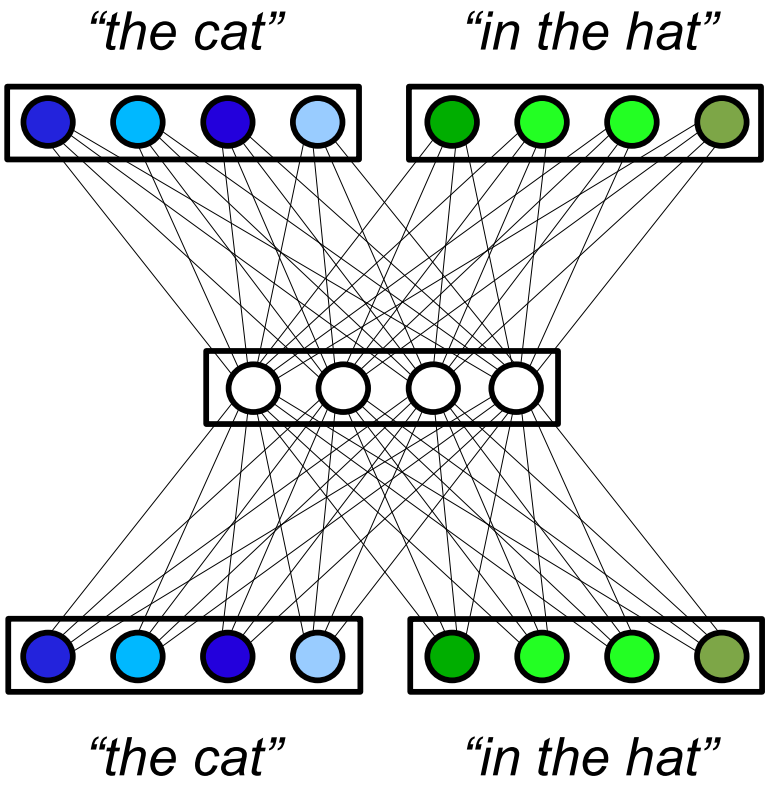
“the” = **1 0 0 0**
“cat” = **0 1 0 0**
“in” = **0 0 1 0**
“hat” = **0 0 0 1**



“the cat” =
“the hat” =
“in the hat” =

RAAM: Recursive Auto-Associative Memory

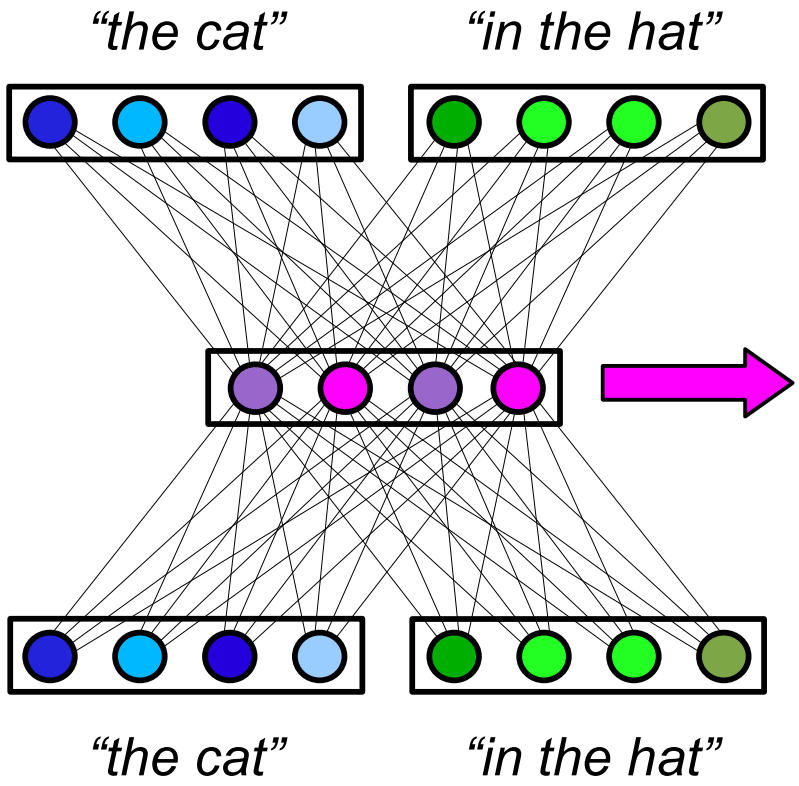
“the” = **1 0 0 0**
“cat” = **0 1 0 0**
“in” = **0 0 1 0**
“hat” = **0 0 0 1**



“the cat” =
“the hat” =
“in the hat” =

RAAM: Recursive Auto-Associative Memory

"the" = **1 0 0 0**
"cat" = **0 1 0 0**
"in" = **0 0 1 0**
"hat" = **0 0 0 1**

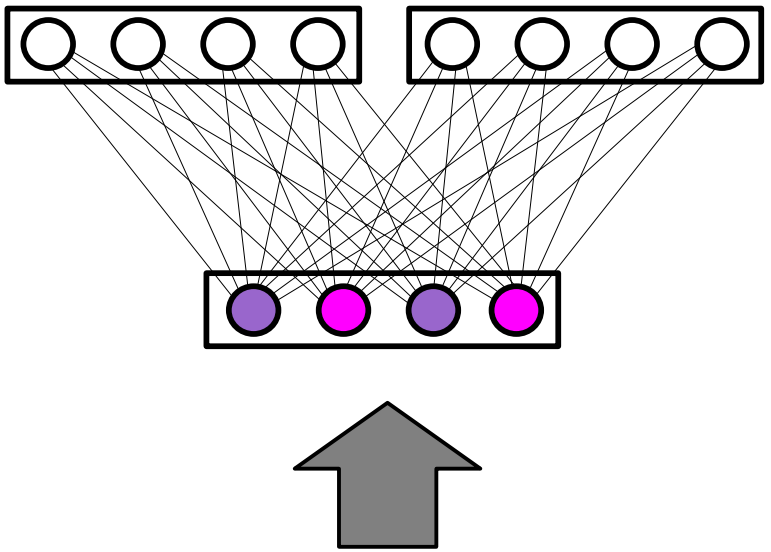


"the cat" =
"the hat" =
"in the hat" =

"the cat in the hat"

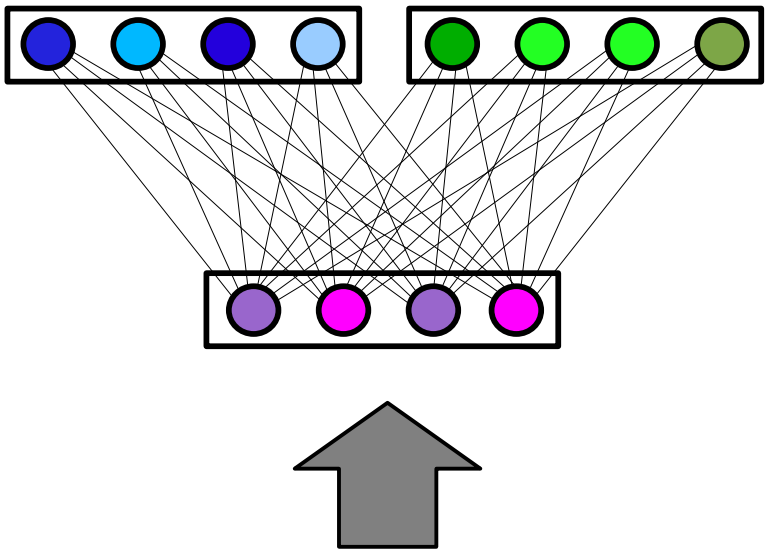
[0.76 0.43 0.81 0.52]

RAAM: Decoding



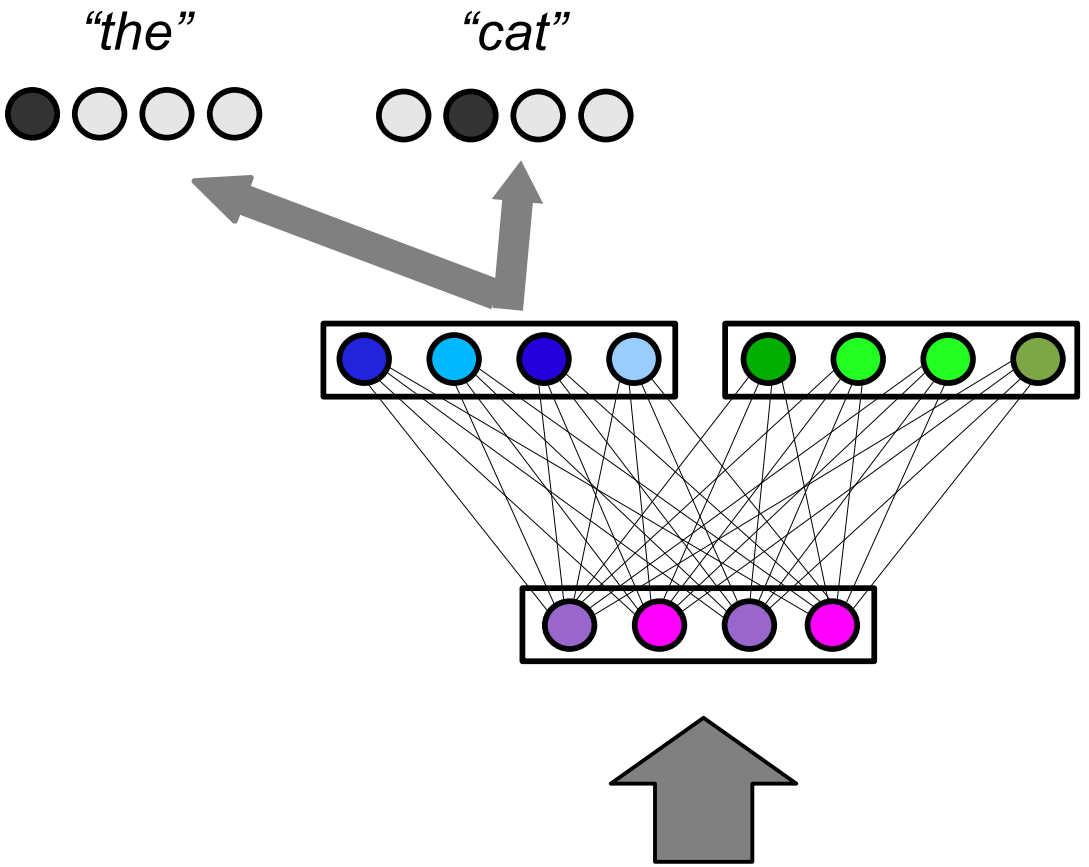
Input = [0.76 0.43 0.81 0.52]

RAAM: Decoding



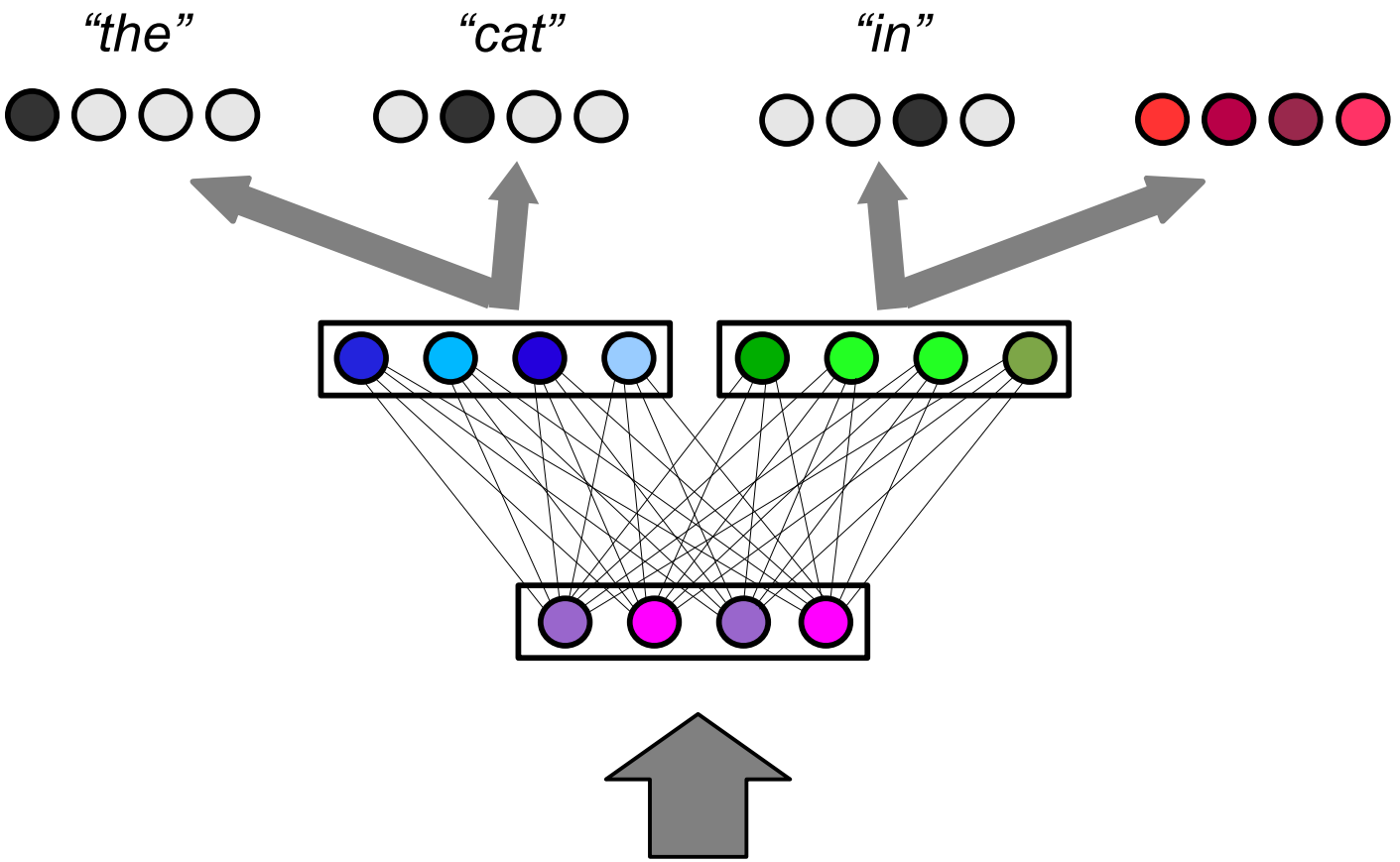
Input = [0.76 0.43 0.81 0.52]

RAAM: Decoding



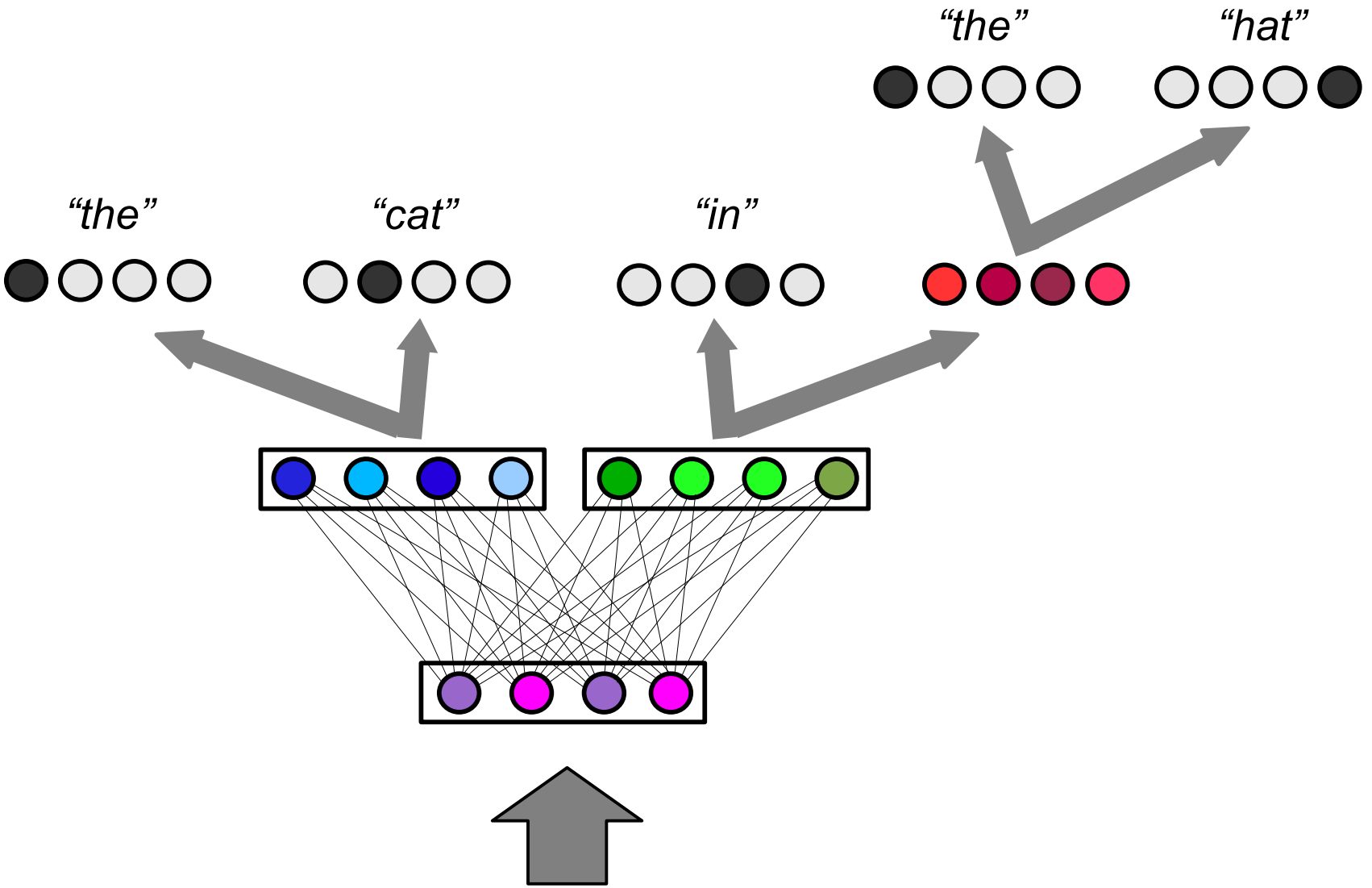
Input = [0.76 0.43 0.81 0.52]

RAAM: Decoding



Input = [0.76 0.43 0.81 0.52]

RAAM: Decoding



Input = [0.76 0.43 0.81 0.52]

A Case Study of RAAM

- Joint work with my colleagues Doug Blank and Lisa Meeden
- Used a RAAM to encode simple 2- or 3-word sentences
- Analyzed the resulting distributed representations of words and sentences learned by the network
- Examined the ability to perform computations holistically with distributed representations of sentences
- Used a variation of RAAM called a **sequential RAAM**
- Similar in spirit to David Chalmers' experiments with sentence passivization described in Chapter 7 of *Artificial Minds*

A Case Study of RAAM

- A grammar for generating simple 2- or 3-word “sentences”
- 26 possible words, 1 special end-marker symbol
- 15 nouns:
  tarzan, jane, boy, cheetah, chimp, rhino, bigfoot, junglebeast, coconut, banana, berries, meat, tree, rock, jeep
- 11 verbs:
  flee, hunt, kill, chase, squish, move, eat, see, smell, exist, swing
- Sample sentences:
  tarzan see chimp
  jane hunt
  cheetah chase boy

Lexical Categories

Category	Members
NOUN-ANIMATE	*tarzan jane boy cheetah chimp rhino*
NOUN-AGGRESSIVE	*cheetah rhino bigfoot junglebeast*
NOUN-EDIBLE	*coconut banana berries meat*
NOUN-SQUISH	*banana berries*
NOUN-MOBILE	NOUN-ANIMATE + *(bigfoot junglebeast jeep)*
NOUN-SWINGER	*tarzan chimp*
NOUN-HUNTER	*jane*
NOUN	NOUN-ANIMATE + *(bigfoot junglebeast)* + NOUN-EDIBLE + *(jeep tree rock)*
NOUN-REAL	NOUN − *(bigfoot junglebeast)*
VERB-FLEE	*flee*
VERB-HUNT	*hunt*
VERB-AGGRESS	*kill chase*
VERB-SQUISH	*squish*
VERB-MOVE	*move*
VERB-EAT	*eat*
VERB-PERCEIVE	*see smell*
VERB-INTRANS	*see smell*
VERB-EXIST	*exist*
VERB-SWING	*swing*

Sentence Templates

Template	Word1	Word2	Word3
1	NOUN-ANIMATE	VERB-FLEE	NOUN-AGGRESSIVE
2	NOUN-AGGRESSIVE	VERB-AGGRESS	NOUN-ANIMATE
3	NOUN-ANIMATE	VERB-SQUISH	NOUN-SQUISH
4	NOUN-ANIMATE	VERB-EAT	NOUN-EDIBLE
5	NOUN-ANIMATE	VERB-PERCEIVE	NOUN
6	NOUN-MOBILE	VERB-MOVE	
7	NOUN-ANIMATE	VERB-INTRANS	
8	NOUN-REAL	VERB-EXIST	
9	NOUN-SWINGER	VERB-SWING	
10	NOUN-HUNTER	VERB-HUNT	
11	NOUN-AGGRESSIVE	VERB-HUNT	

Template	Example sentence
1	jane flee junglebeast
2	cheetah kill chimp
3	boy squish banana
4	rhino eat meat
5	bigfoot see jeep
6	tarzan move
7	chimp smell
8	tree exist
9	tarzan swing
10	jane hunt
11	junglebeast hunt

Semantic Constraints

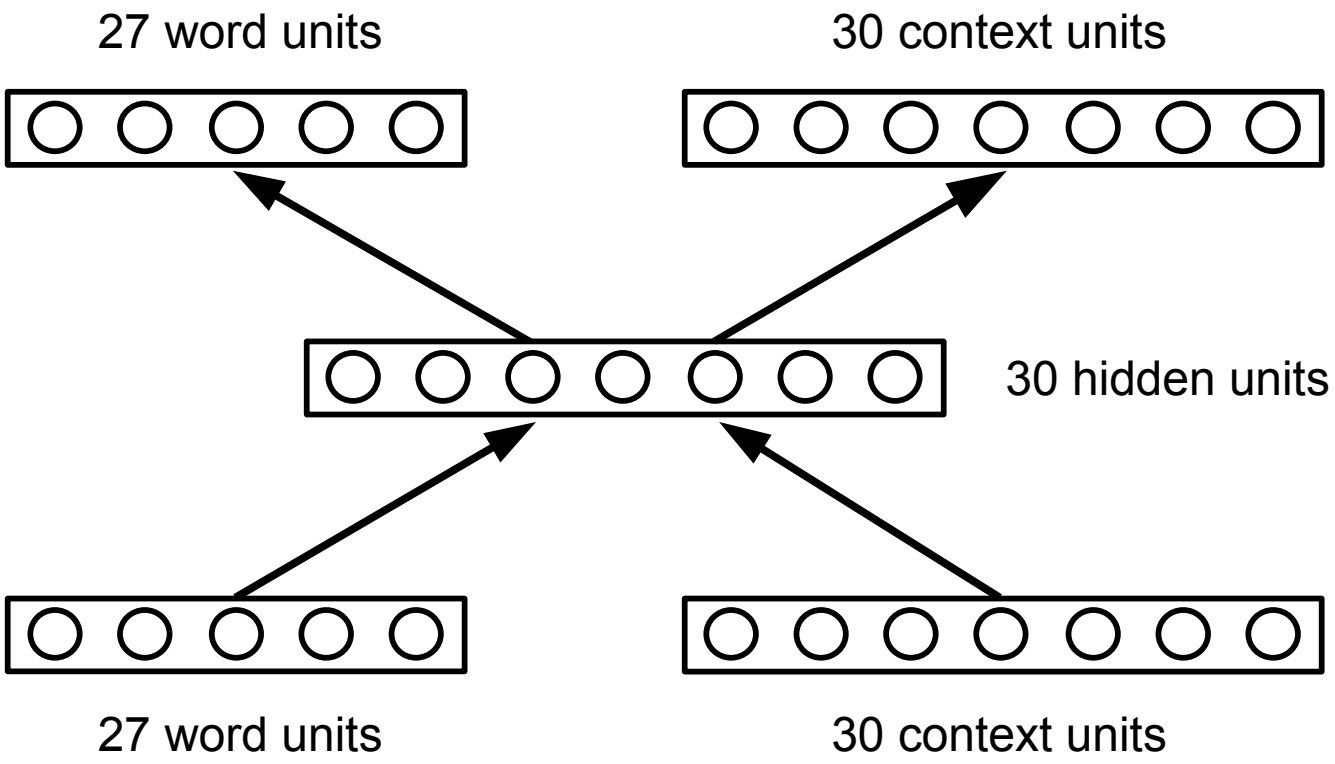
- Grammar highly constrains the set of valid sentences
  - 18,252 possible 2- or 3-word sequences (= $26^2 + 26^3$)
  - only 341 valid sentences
- These constraints reflect the **semantics** of words
- Examples of **ungrammatical** sentences:
  - *banana chase junglebeast*
  - *tarzan eat rock*
  - *coconut see jane*
  - *bigfoot exist*
  - *squish move rhino*

Word Representations

- 27-bit **localist** representation of words
  - **1**000000000000000000000000000 = *tarzan*
  - **01**00000000000000000000000000 = *jane*
  - **001**0000000000000000000000000 = *boy*

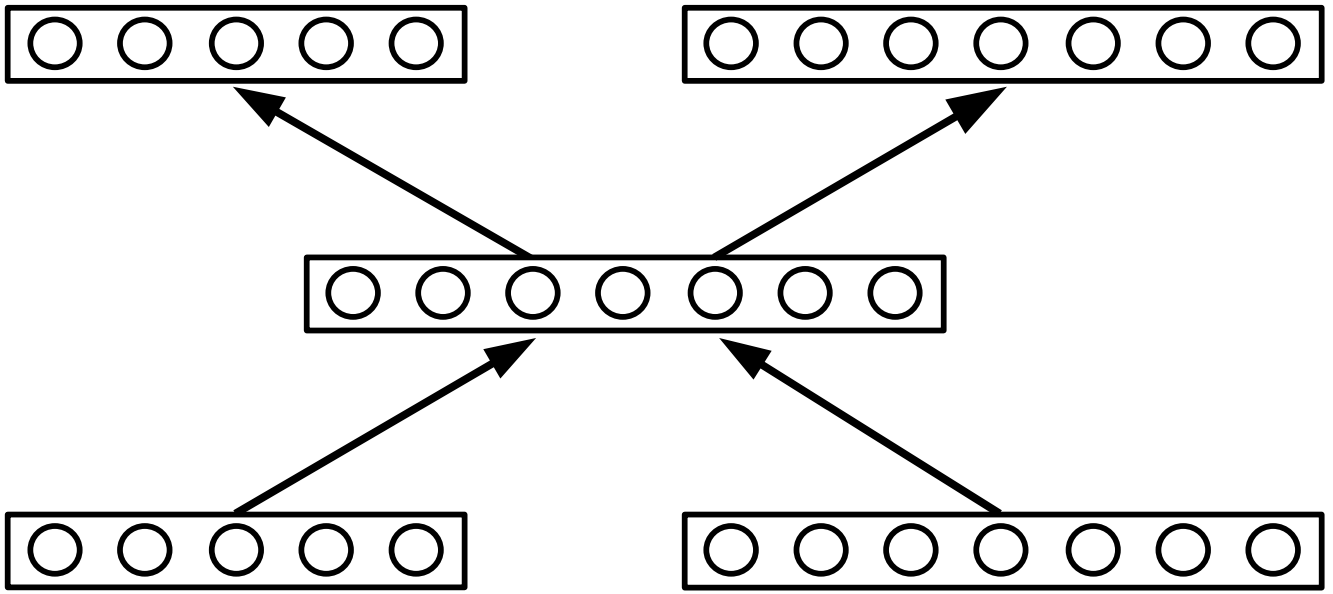
    . . .
  - 000000000000000000000000000**1** = *end-marker*
- This ensures that the representations of individual words have **no intrinsic similarity** to each other
- From the network's point of view, each word is just an **abstract symbol**, with no relation to any other symbol
- Sentences are presented to the network **one word at a time,** with no information about sentence structure (no parse tree)

Sequential RAAM Architecture



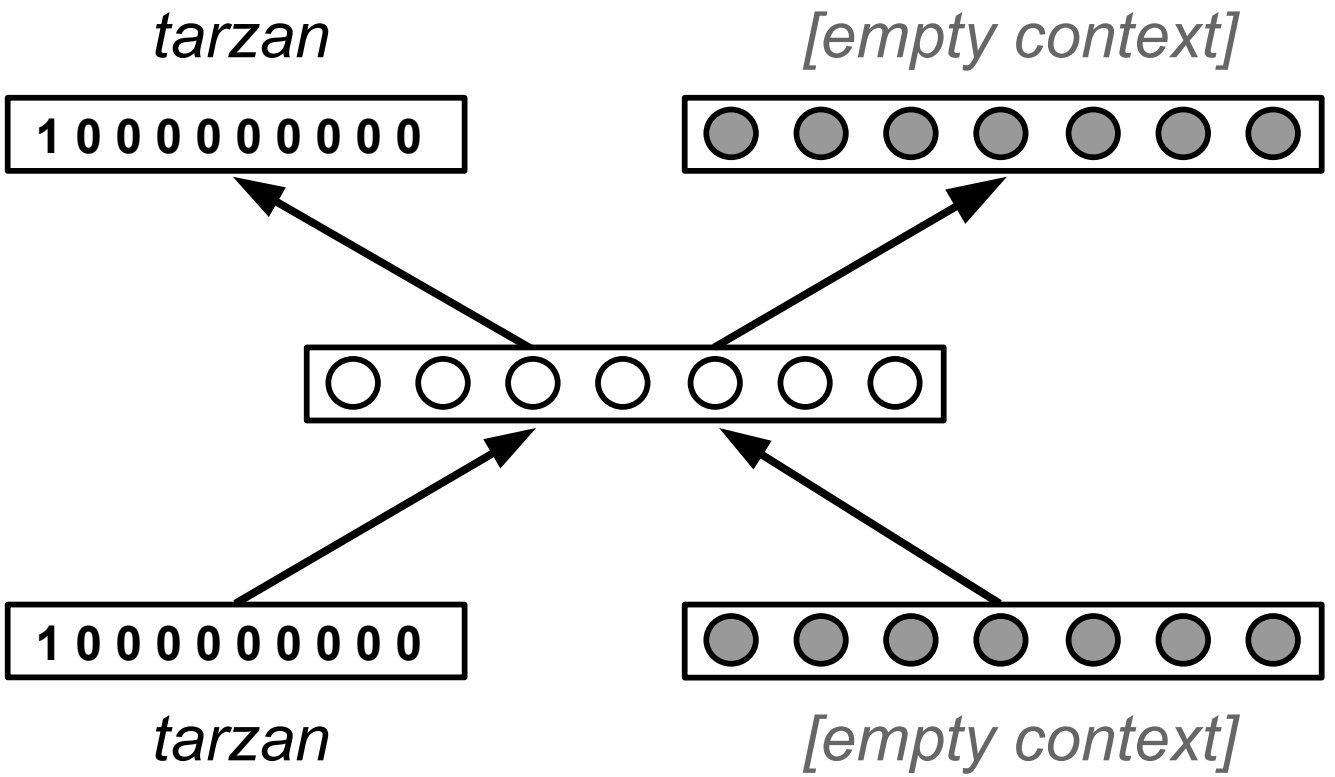
Encoding Sentences

“tarzan see chimp”



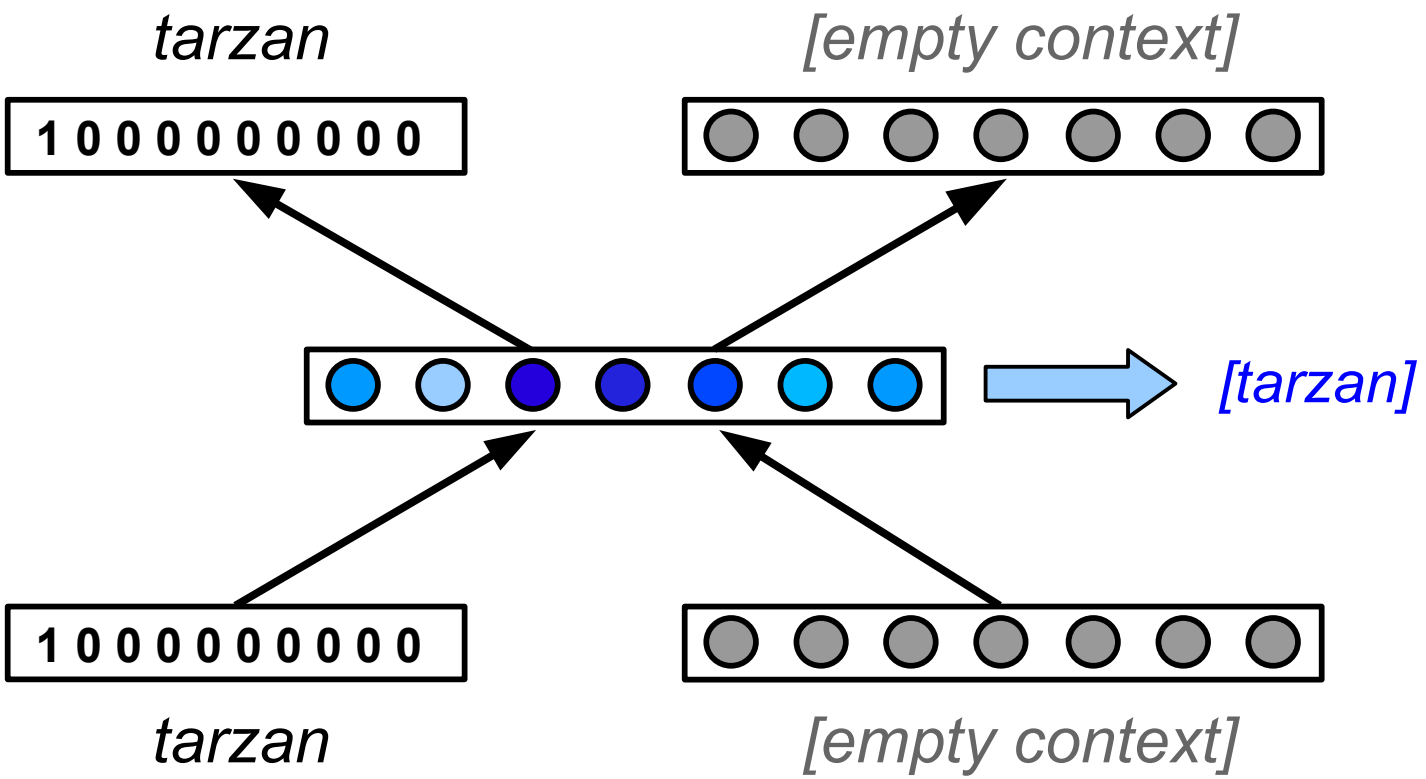
Encoding Sentences

“tarzan see chimp”



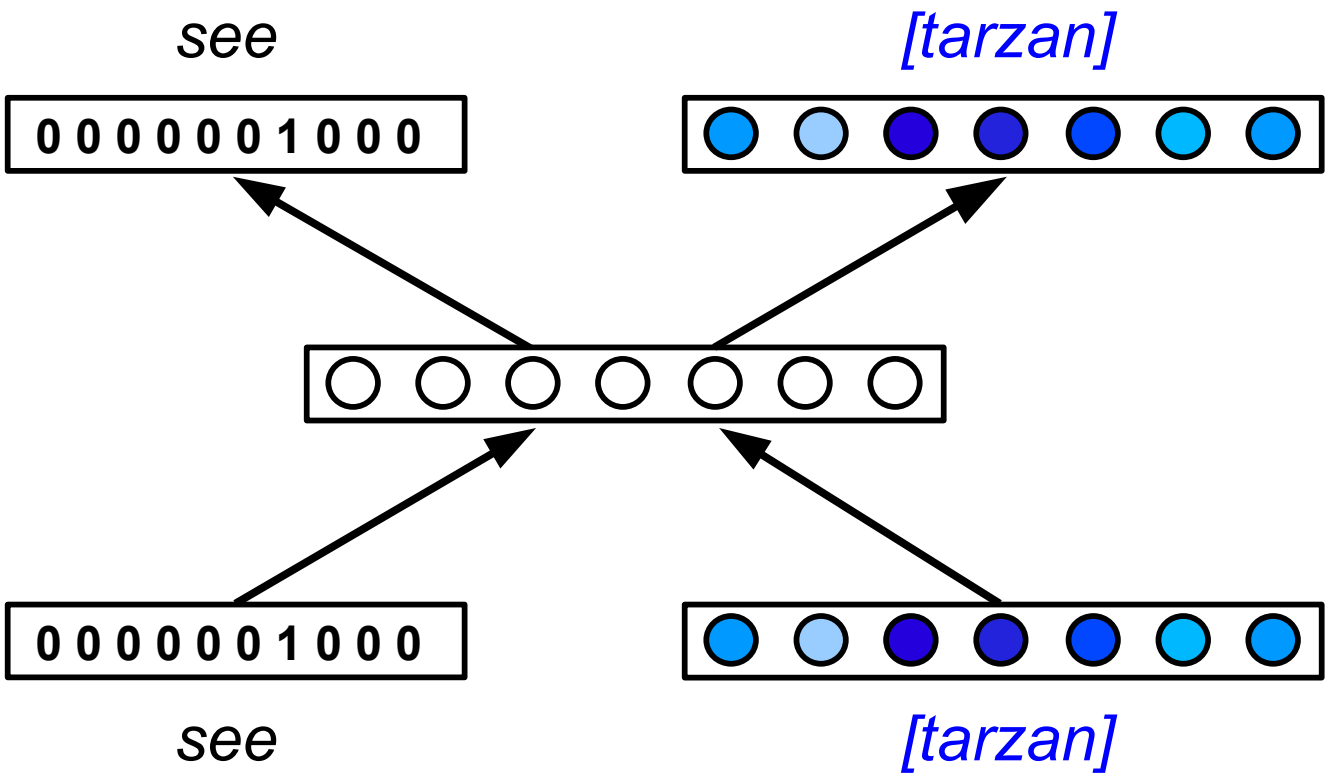
Encoding Sentences

“tarzan see chimp”



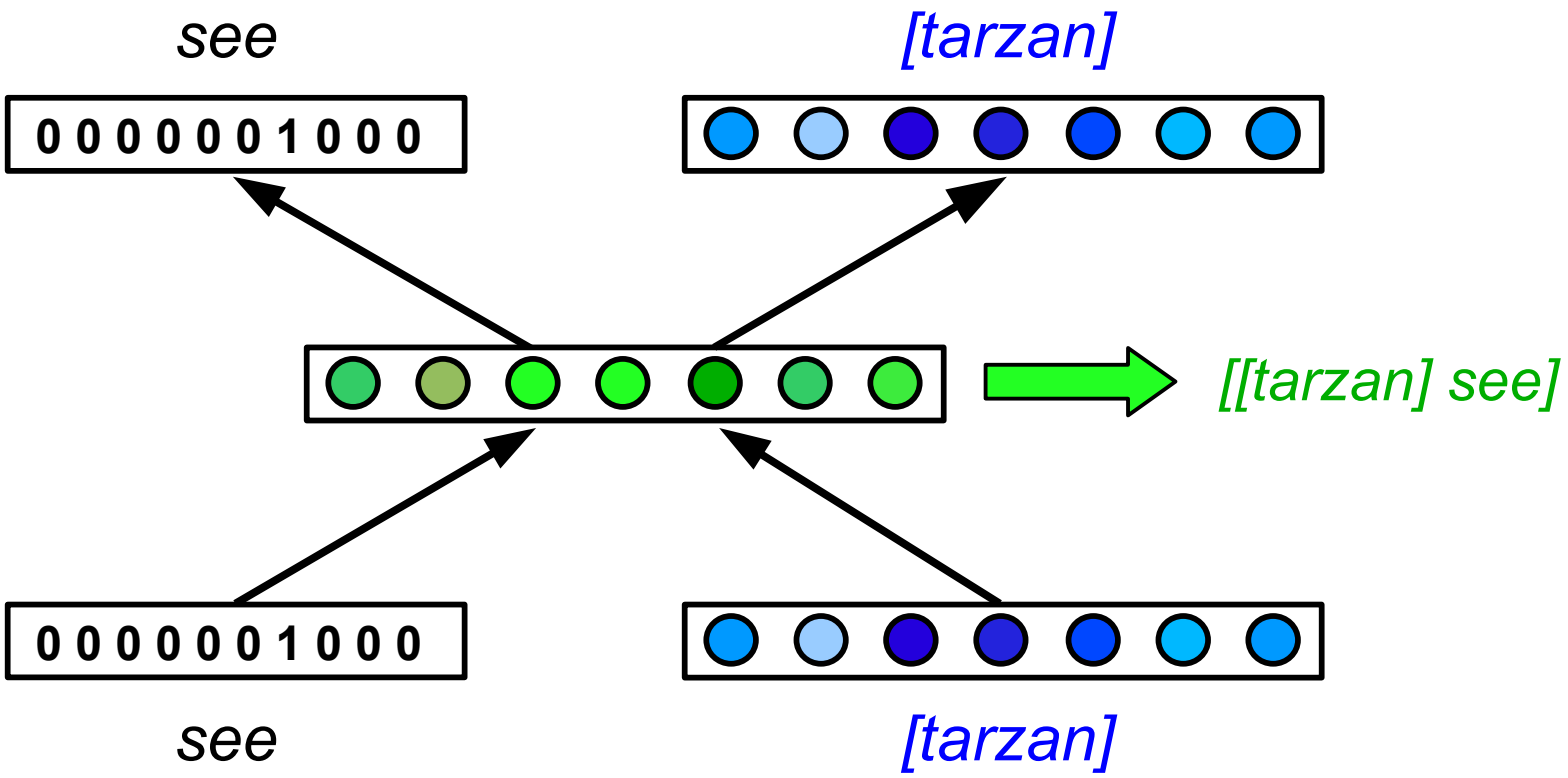
Encoding Sentences

“tarzan see chimp”



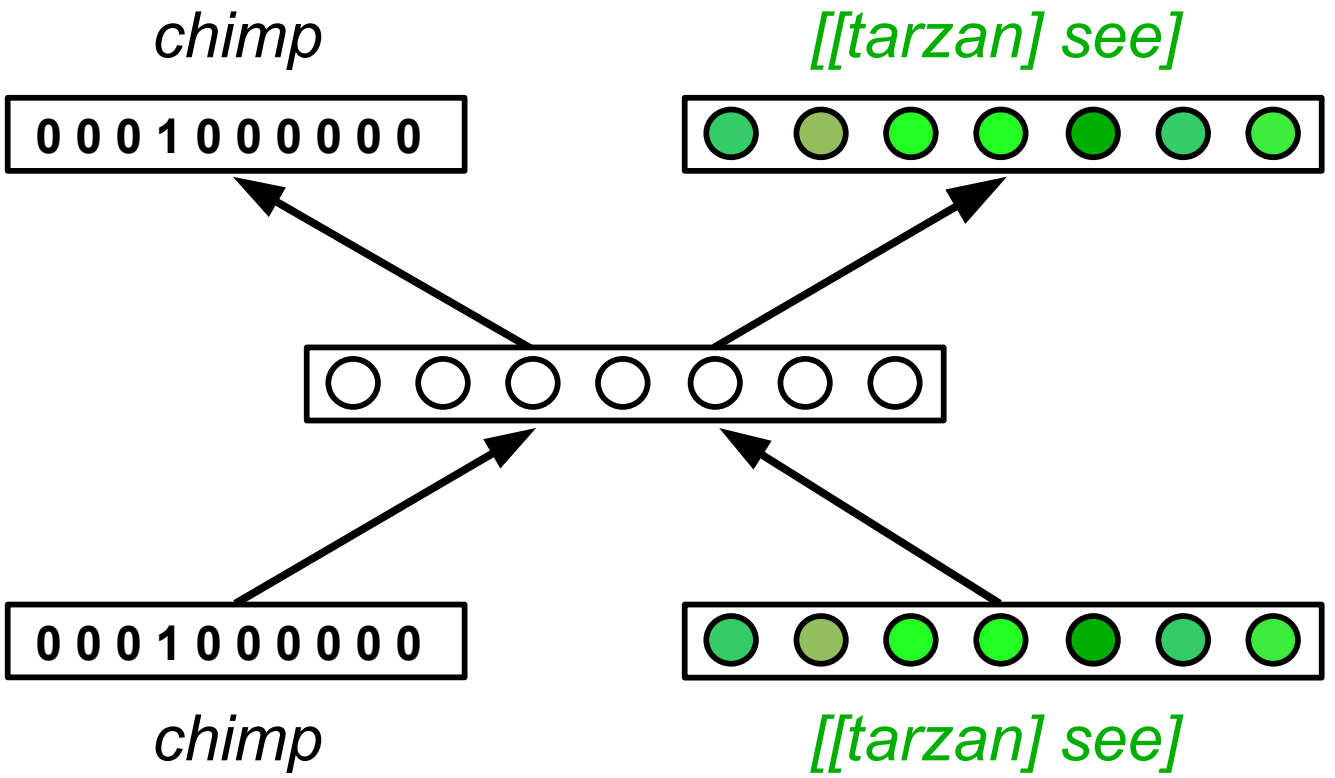
Encoding Sentences

“tarzan see chimp”



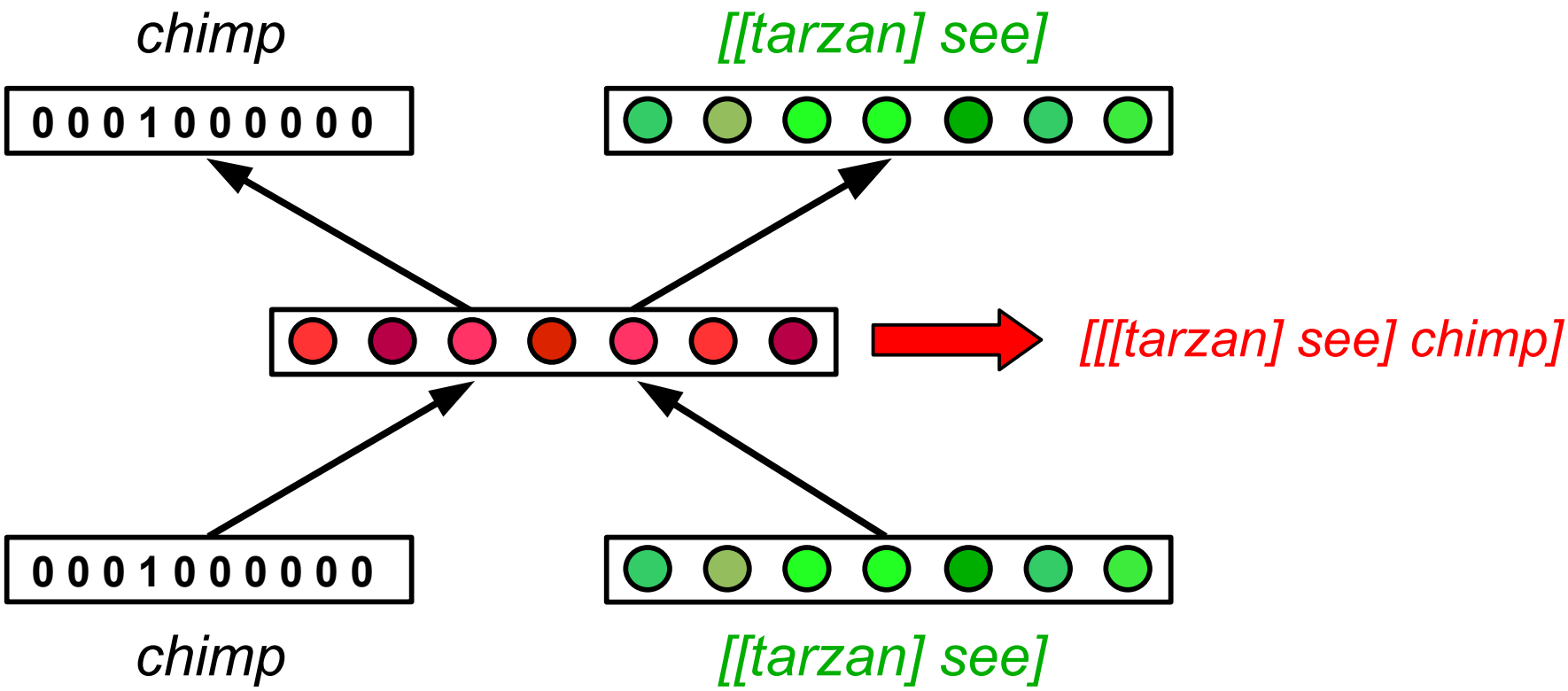
Encoding Sentences

“tarzan see chimp”

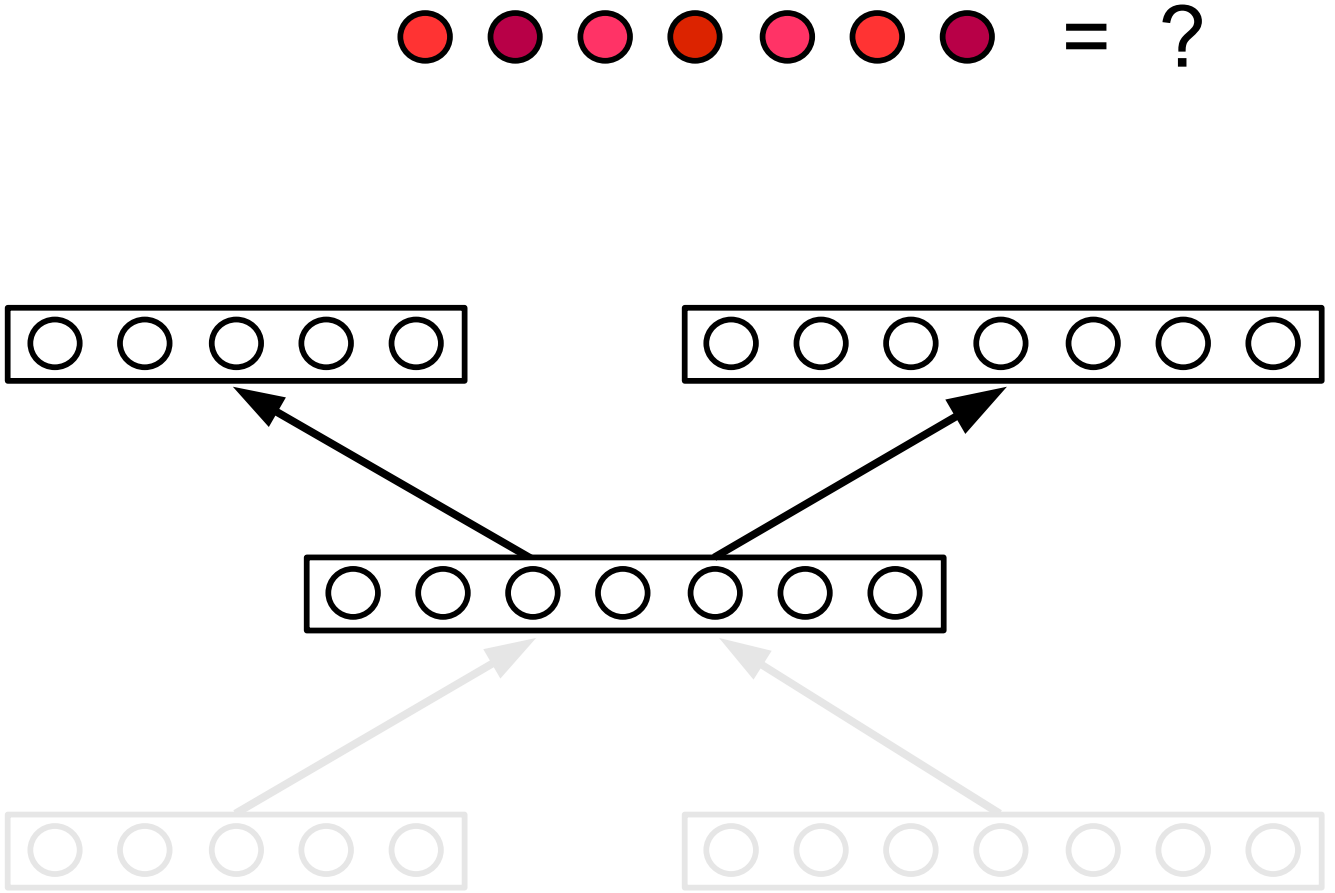


Encoding Sentences

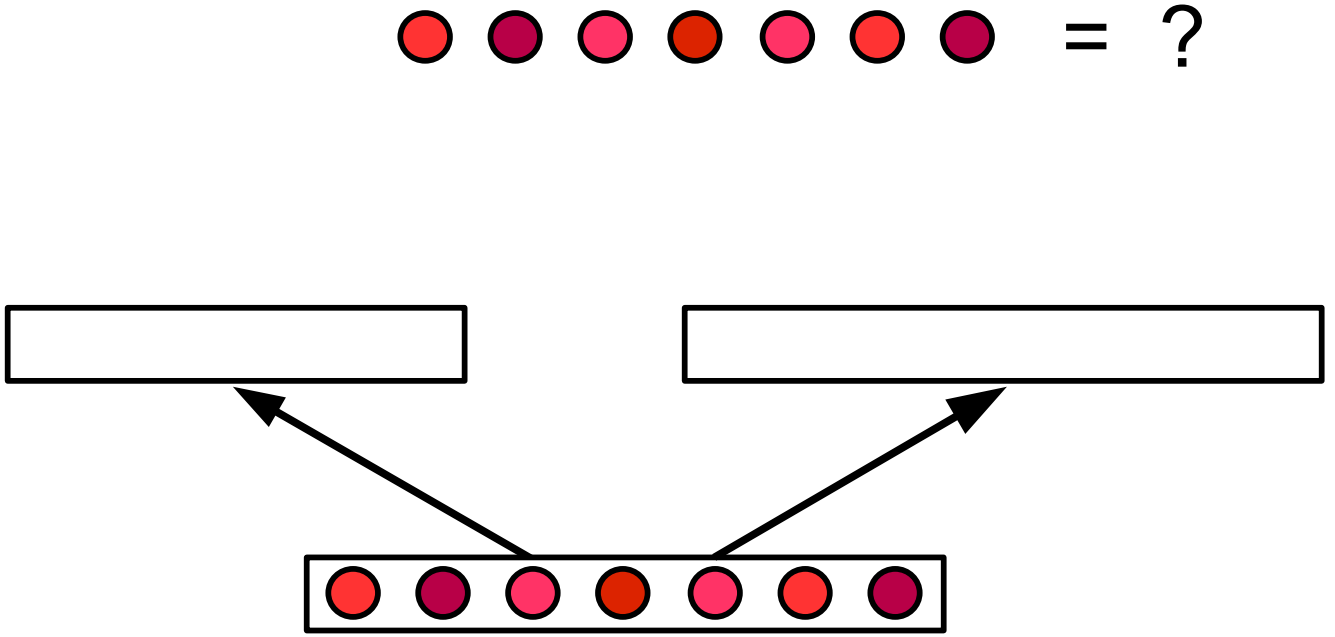
“tarzan see chimp”



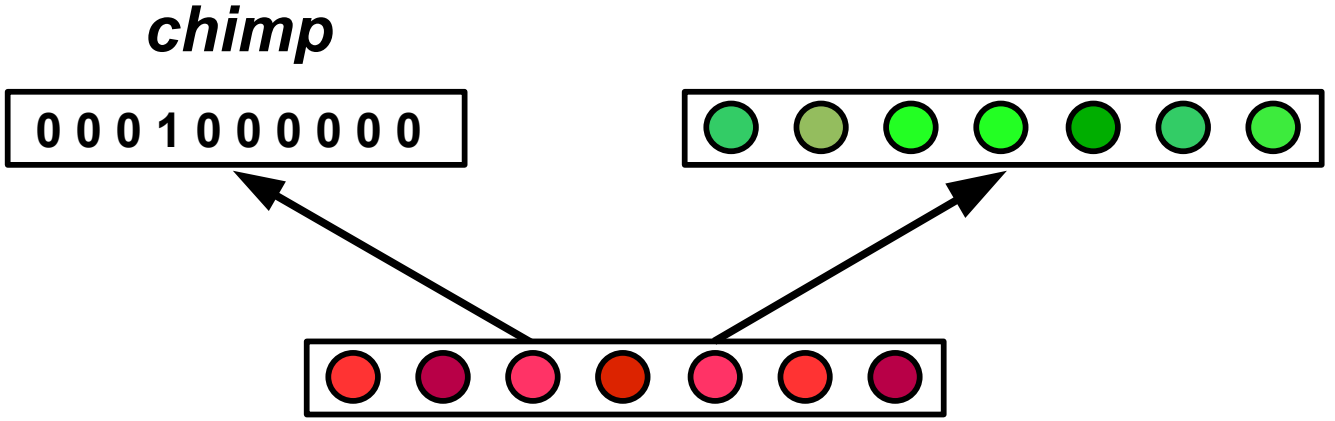
Decoding Sentences



Decoding Sentences

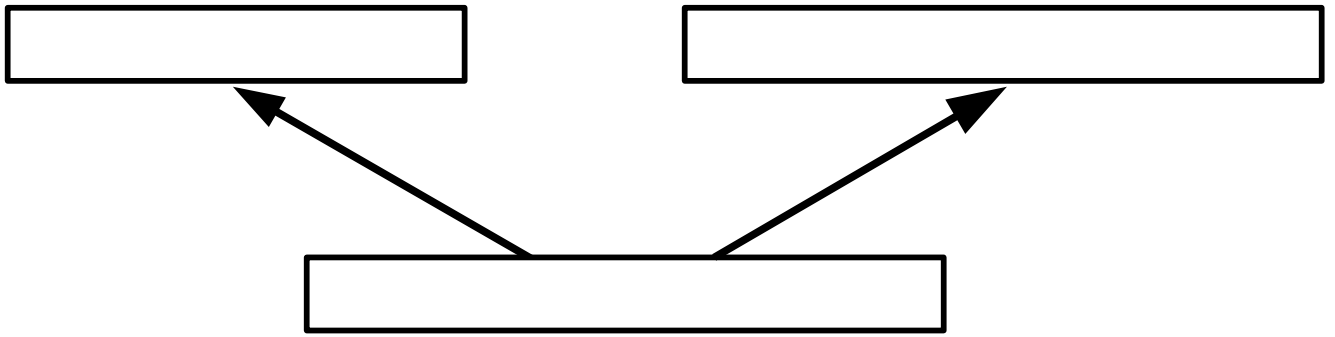


Decoding Sentences



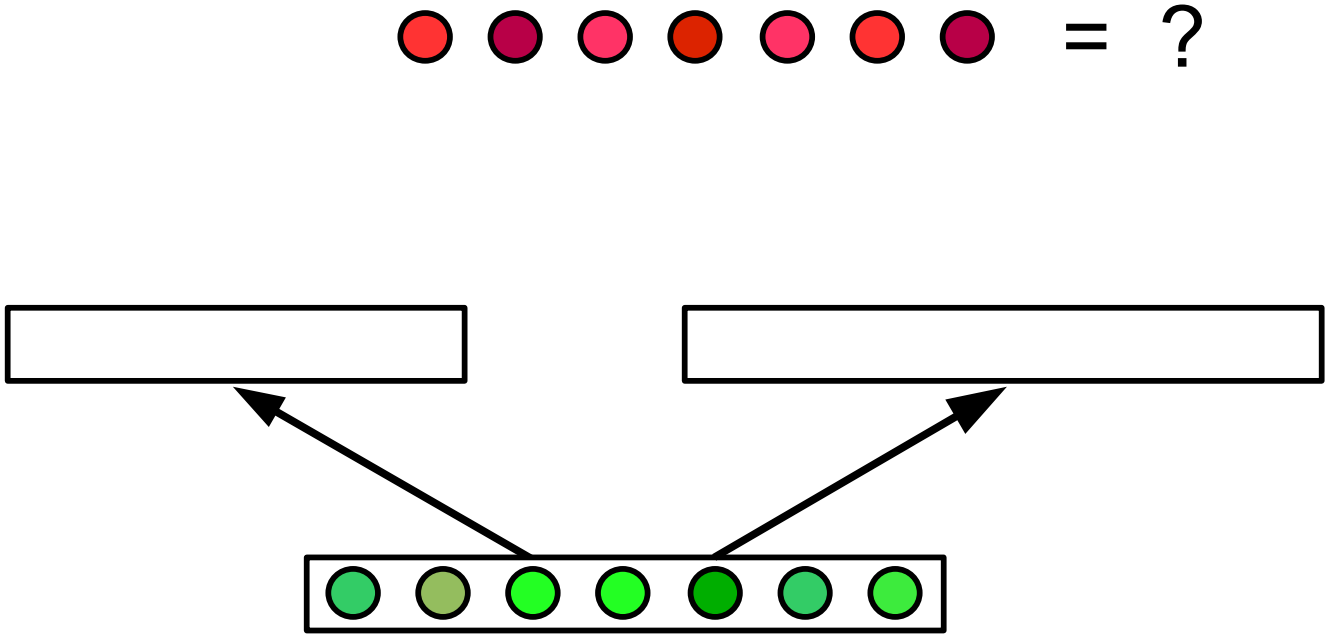
Decoding Sentences

= ?



chimp

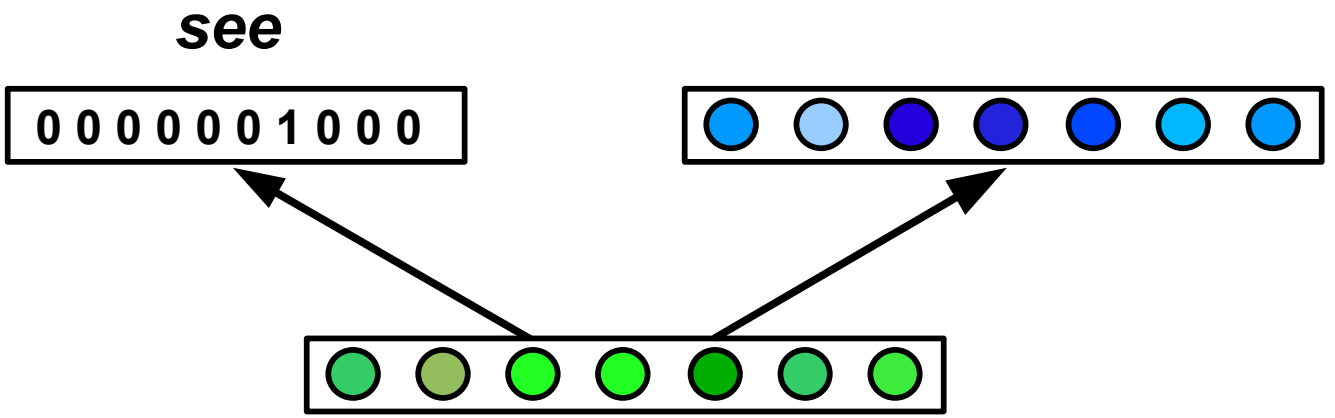
Decoding Sentences



chimp

Decoding Sentences

= ?

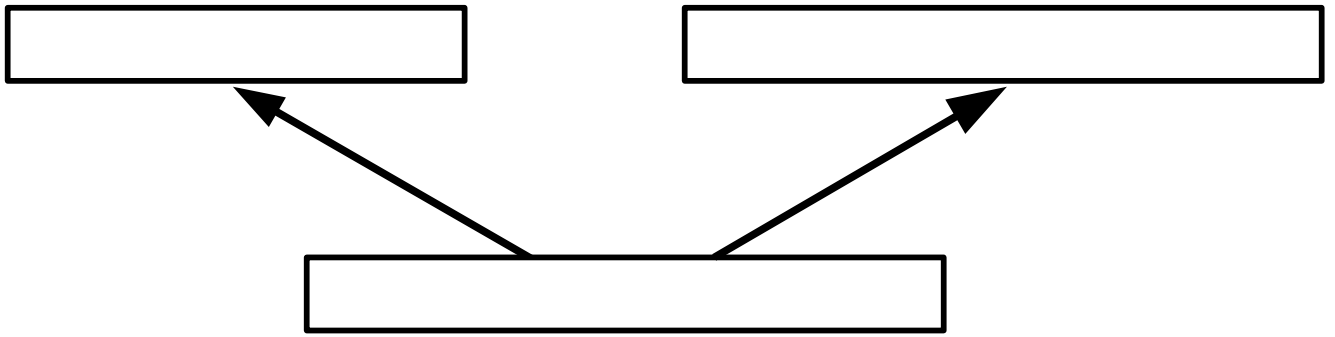


chimp

Decoding Sentences

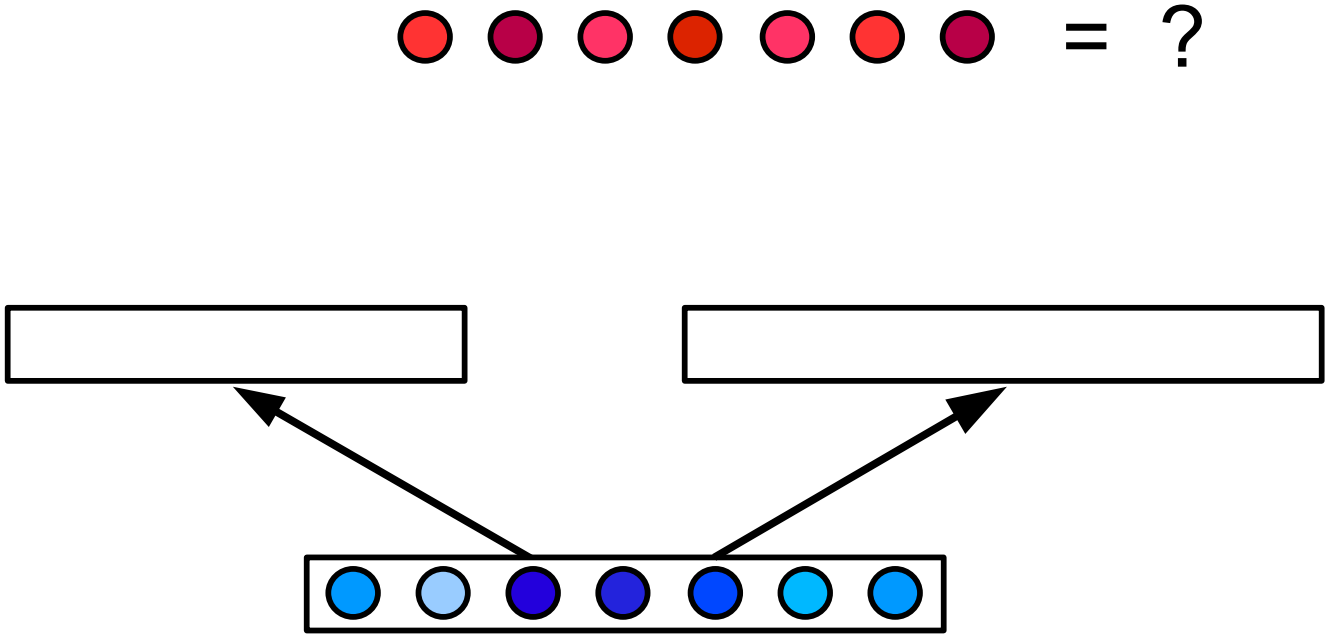
=

?



chimp see

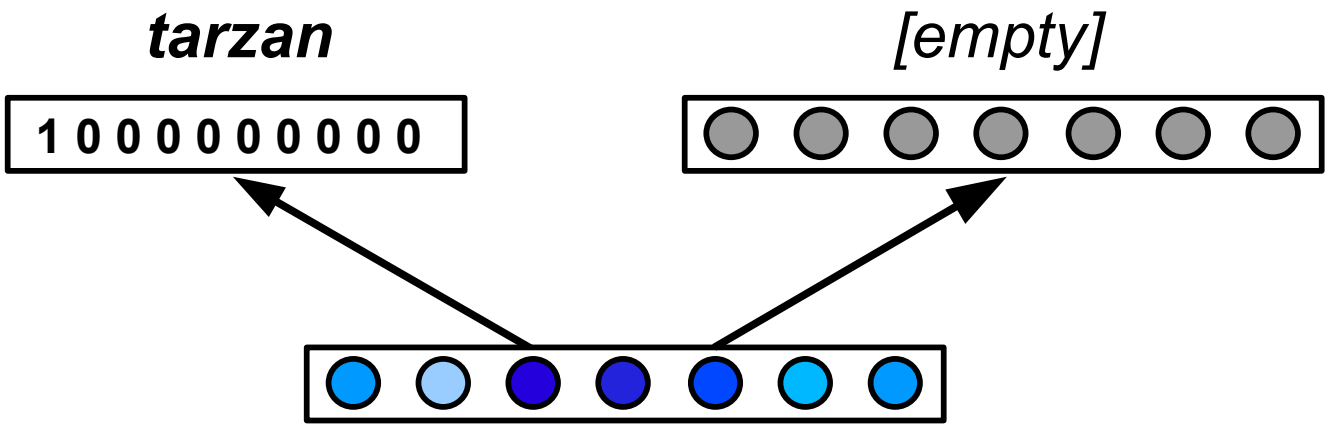
Decoding Sentences



chimp see

Decoding Sentences

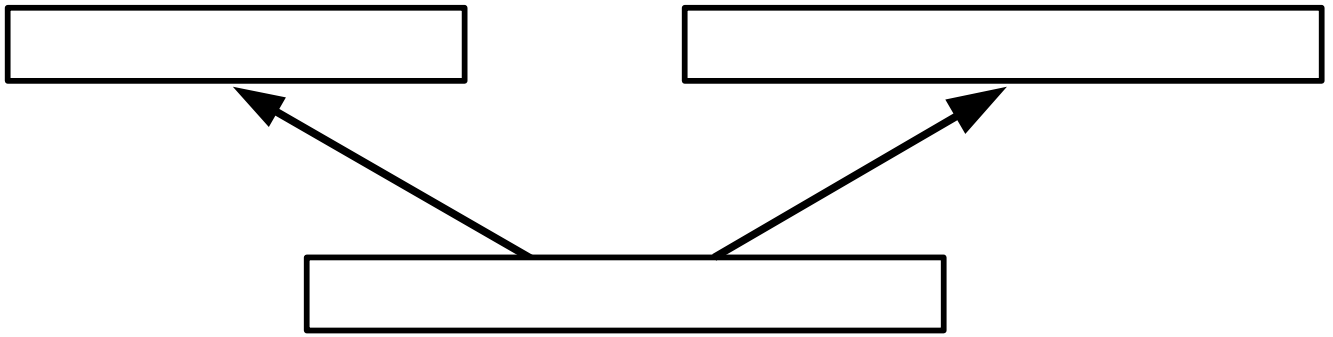
= ?



chimp see

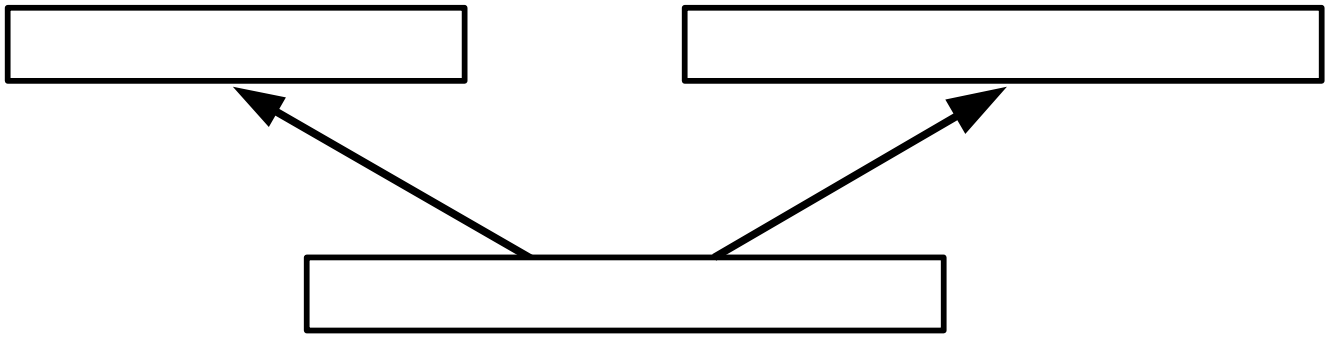
Decoding Sentences

= ?



chimp ***see*** ***tarzan*** *[empty]*

Decoding Sentences



chimp ***see*** ***tarzan*** *[empty]*

“tarzan see chimp”

Experiments

- Test 1: How well can the trained network encode and decode **valid** sentences?
- TRAINING set: 100 randomly-chosen valid sentences
  - presented in random order for ~ 21,000 training cycles
- TESTING set: 100 different valid sentences
  - Task: encode a sentence and then decode it to see if the resulting sentence is the same
  - If the correct word unit is less than 0.5, or is not the most highly activated unit, the network's response is an error
  - 100% of TRAINING sentences passed
  - 80% of TESTING sentences passed
  - Incorrect words were usually still the right grammatical type (*e.g.* the network produced *jane* instead of *jeep*)

Experiments

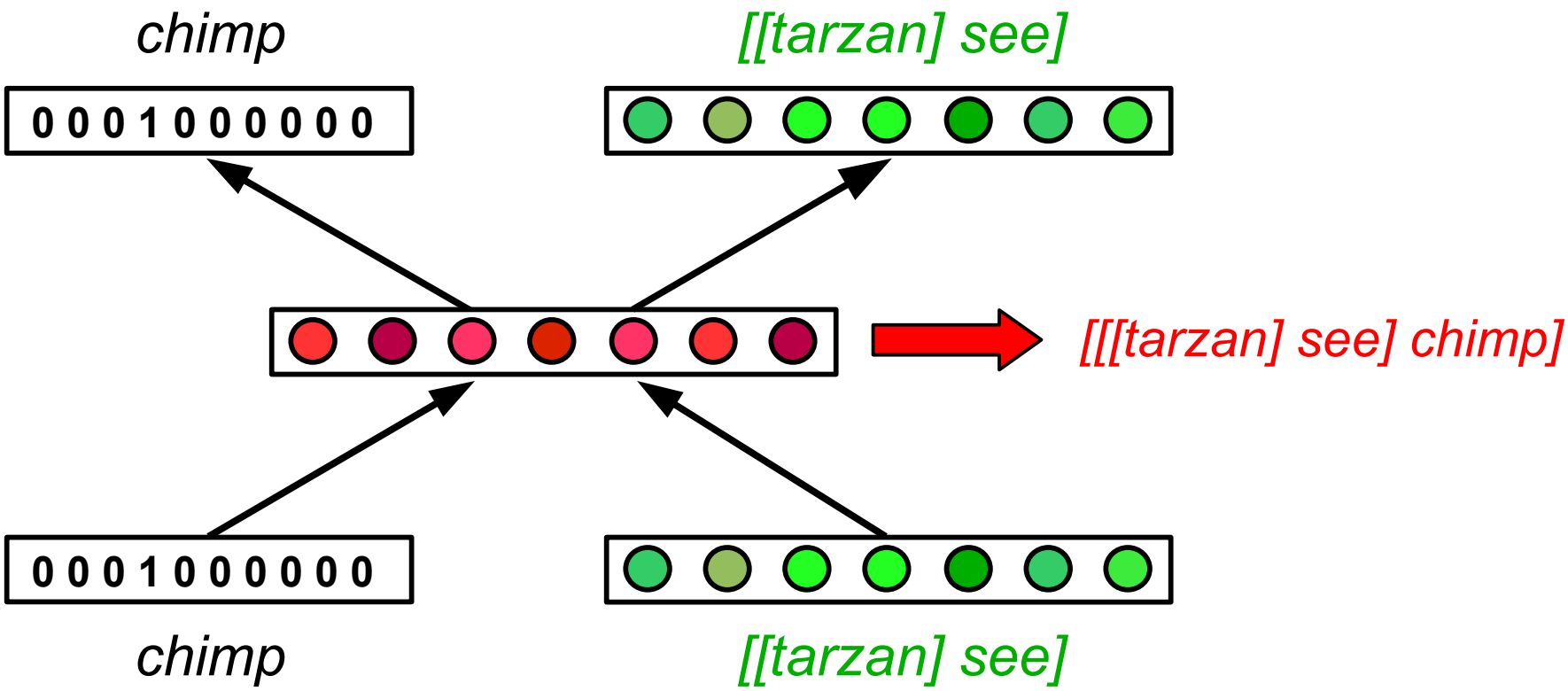
- Test 2: How well can the network encode and decode **ungrammatical** sentences?
- Task: encode and then decode 20 ungrammatical sentences
  - *tarzan chase bigfoot* (*tarzan* not NOUN-AGGRESSIVE)
  - *bigfoot exist* (*bigfoot* not NOUN-REAL)
  - *berries chase meat* (*berries* not NOUN-AGGRESSIVE, *meat* not NOUN-ANIMATE)
  - *eat tree eat* (sentence not of the form NOUN VERB NOUN)
- Only 35% were correctly decoded
  - Of these, 86% were only slightly ungrammatical
- Network had difficulty processing ungrammatical sentences

Experiments

- Test 3: Analyze the **internal representations** of words learned by the network
- Step 1: create a **composite word representation** for each of the 286 words in the TRAINING set sentences
- Step 2: perform a **cluster analysis** of the composite word patterns in order to see their similarity structure
- Remember:
  - no explicit information was ever provided to the network about the grammar, or about the structure of sentences
  - sentences were presented to the network sequentially, as a single unbroken stream of words (like how people experience spoken language)

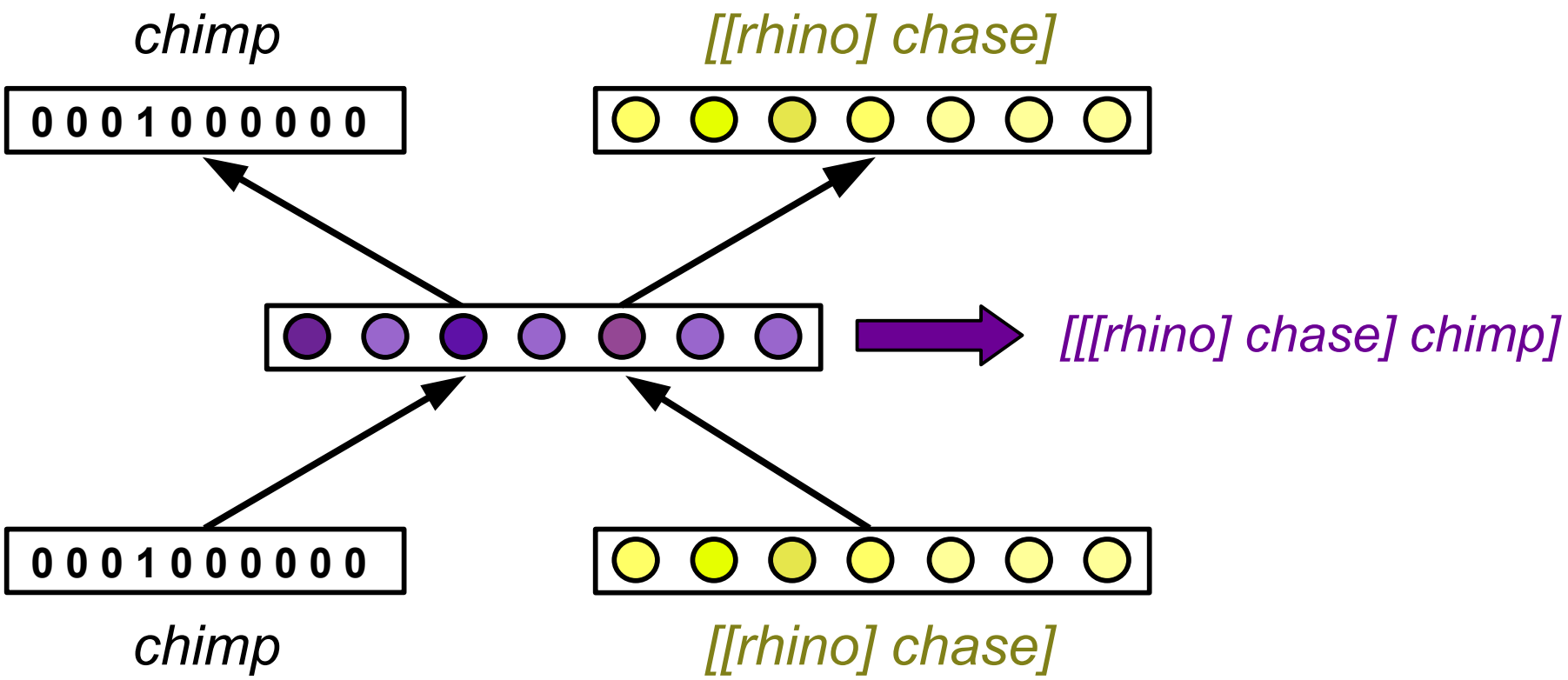
Composite Word Representations

“tarzan see chimp”



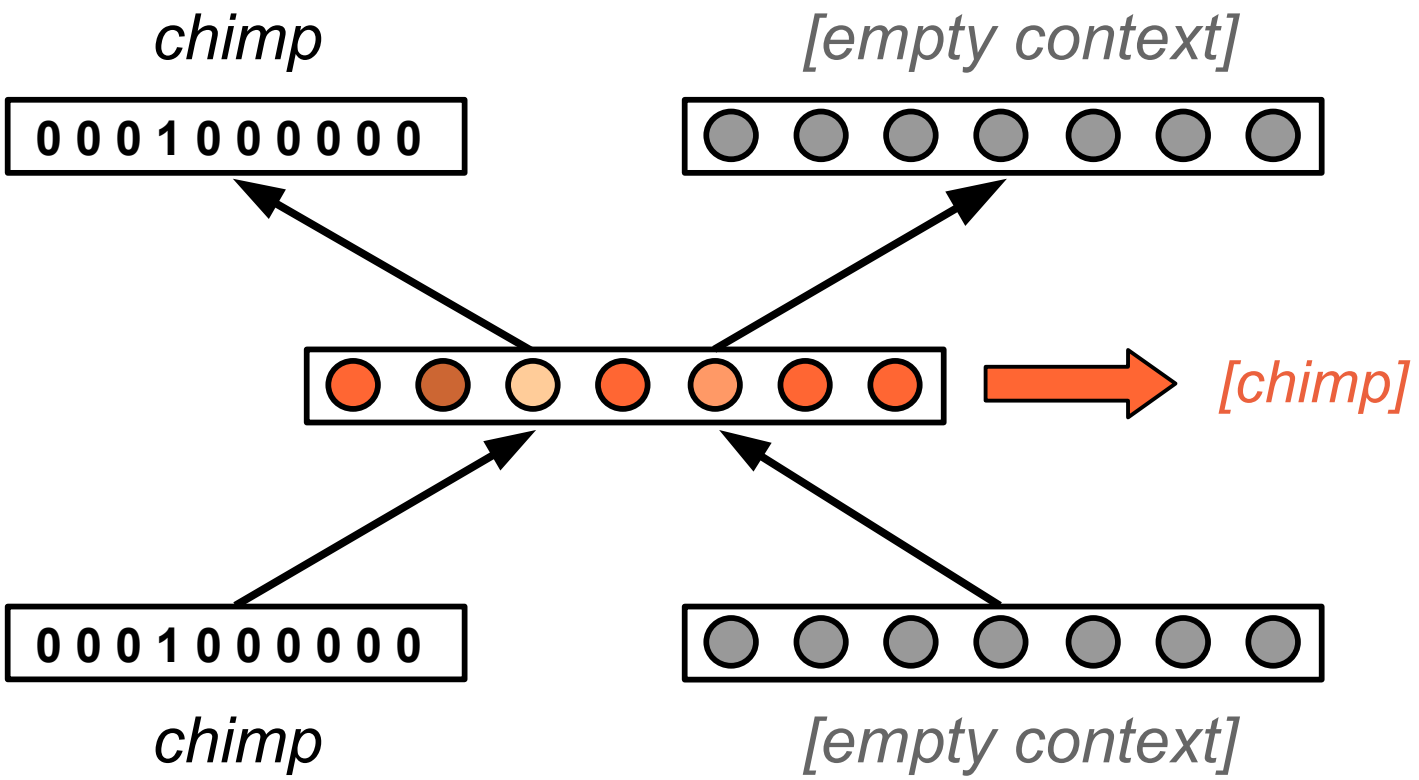
Composite Word Representations

“rhino chase chimp”

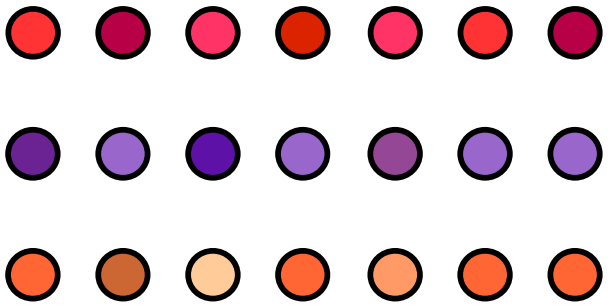


Composite Word Representations

“chimp eat banana”



Composite Word Representations



= *[[[tarzan] see] chimp]*

= *[[[rhino] chase] chimp]*

= *[chimp]*

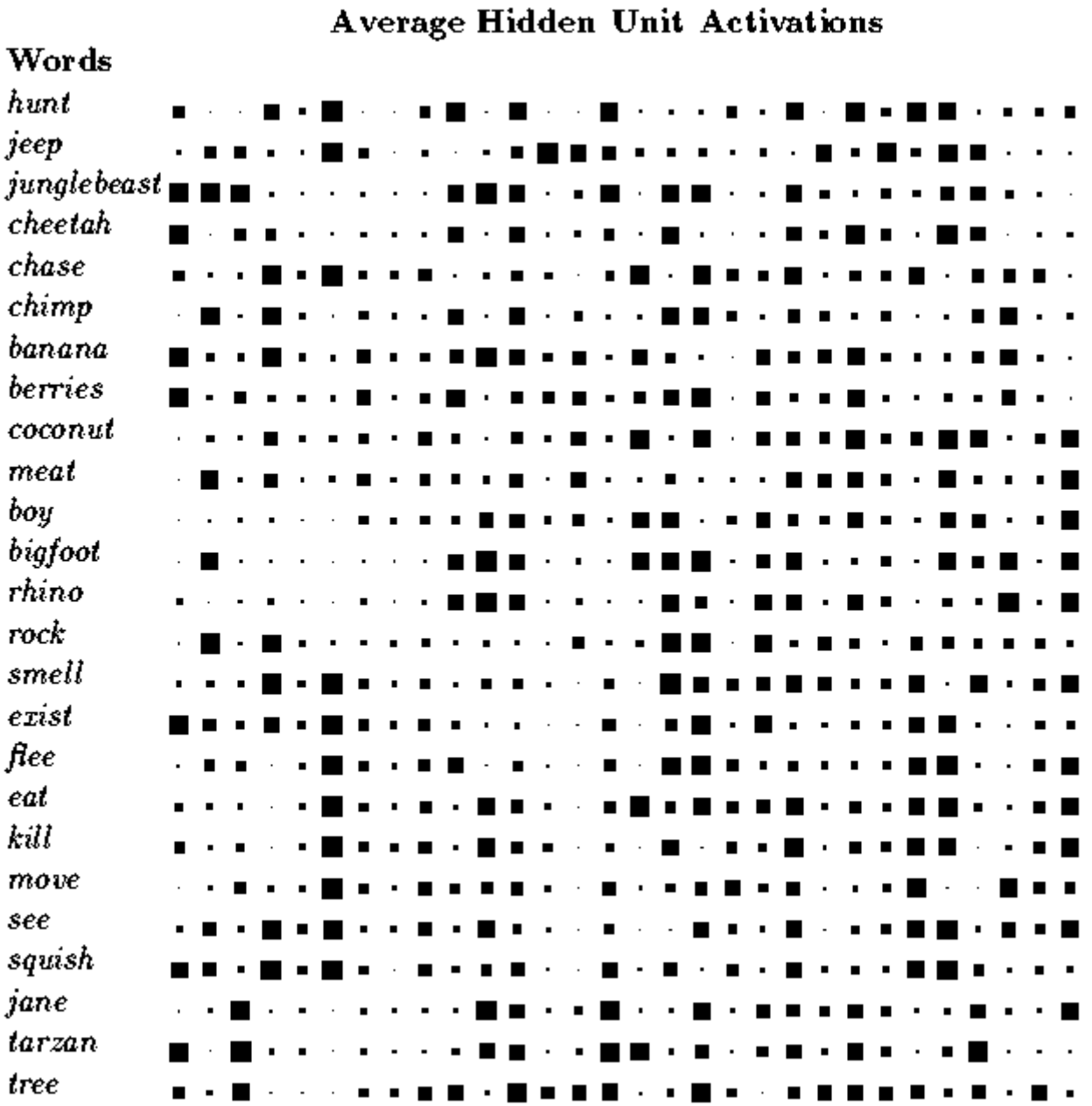
. . .

average = 

= composite word representation of **chimp**

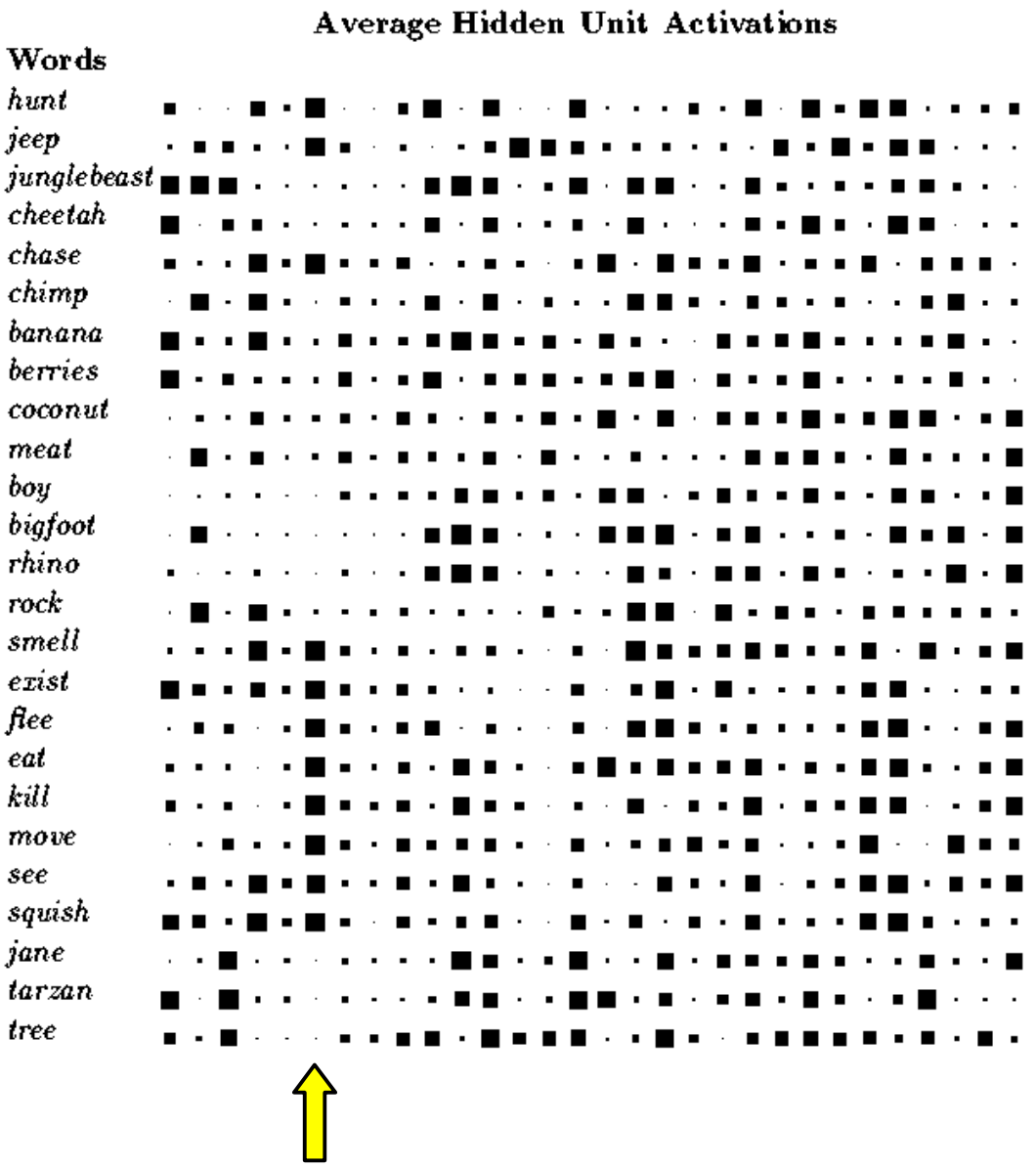
This pattern reflects all of the different contexts in which the word **chimp** is used in the TRAINING set sentences

Composite Word Representations



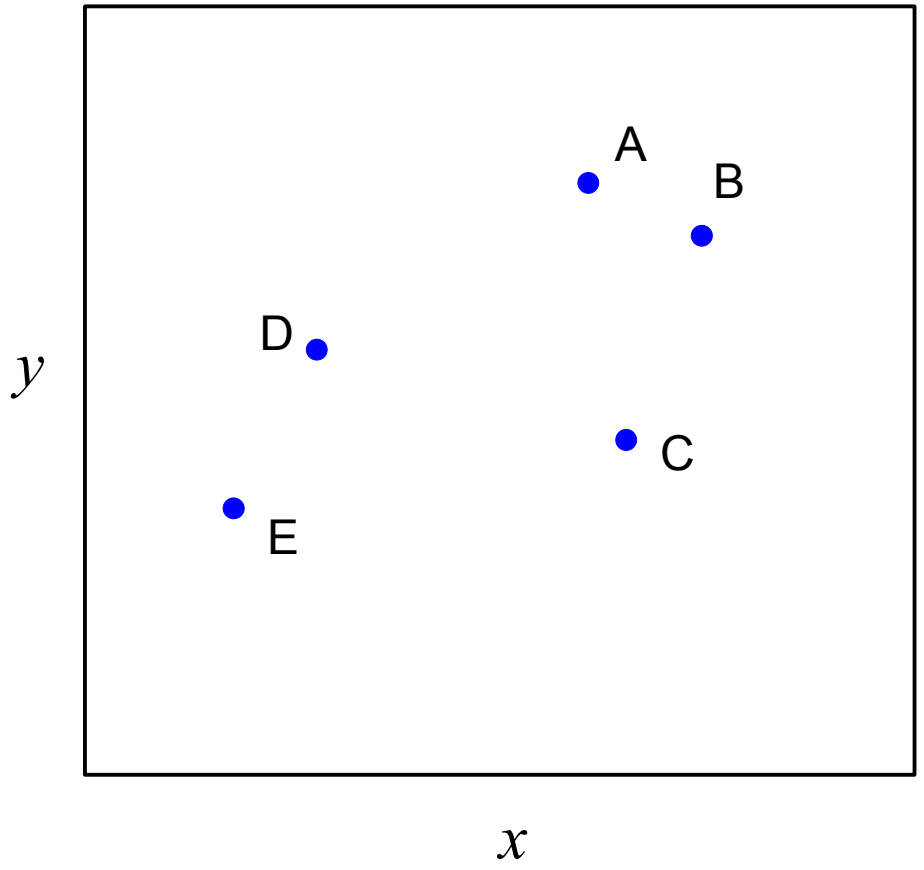
Composite Word Representations

- The network has learned the distinction between nouns and verbs
- 6th pattern value codes for verbs (1) or nouns (0)
- The only exception is *jeep*, which was under-represented in the TRAINING sentences
- A cluster analysis reveals subtler similarities and differences among these word representations



Cluster Analysis

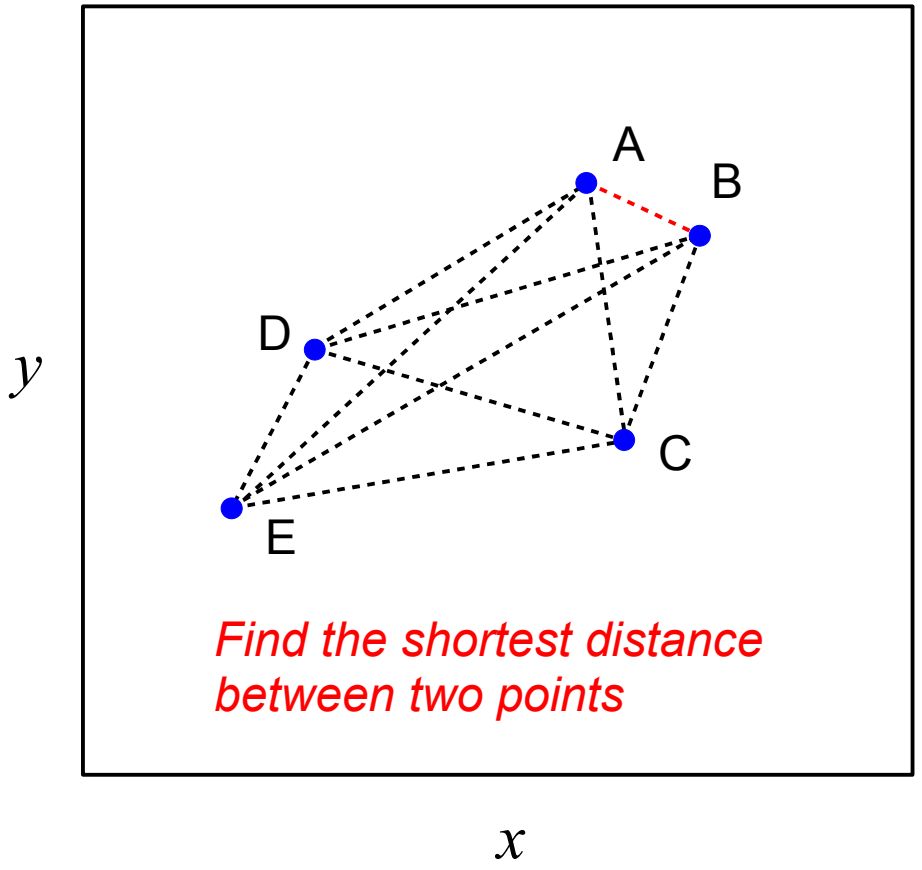
- A geometric method for visualizing the similarity structure of patterns in a multi-dimensional space



	x	y
A	0.69	0.82
B	0.76	0.77
C	0.71	0.43
D	0.28	0.55
E	0.17	0.32

Cluster Analysis

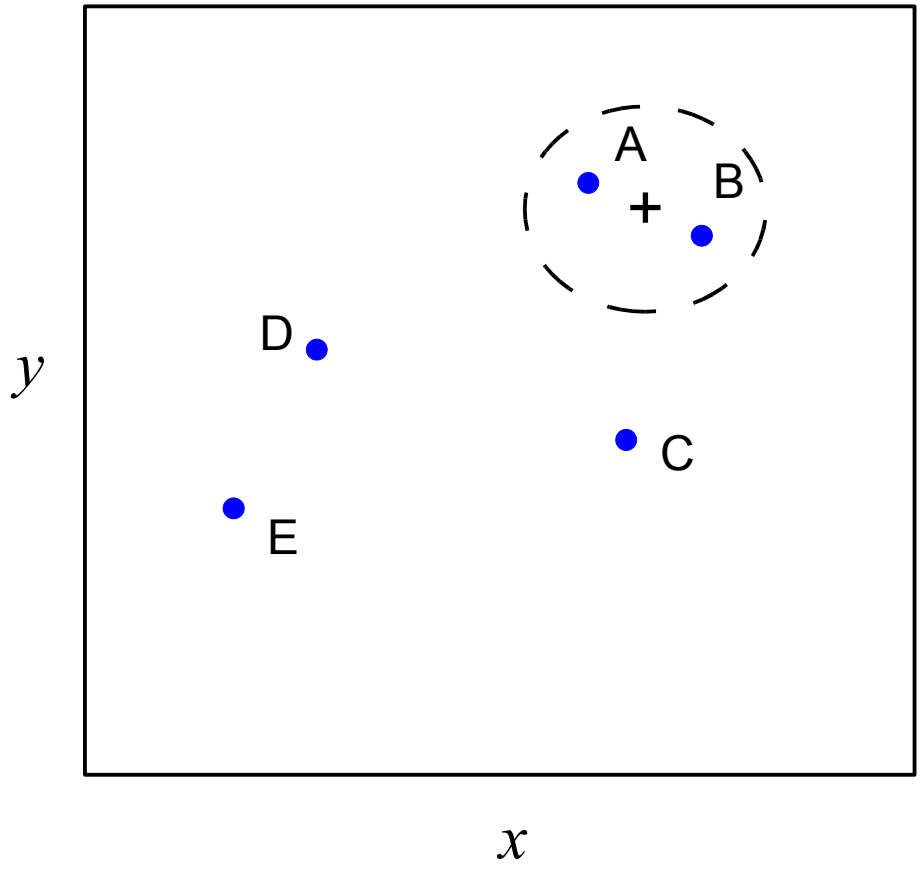
- A geometric method for visualizing the similarity structure of patterns in a multi-dimensional space



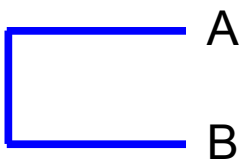
	x	y
A	0.69	0.82
B	0.76	0.77
C	0.71	0.43
D	0.28	0.55
E	0.17	0.32

Cluster Analysis

- A geometric method for visualizing the similarity structure of patterns in a multi-dimensional space

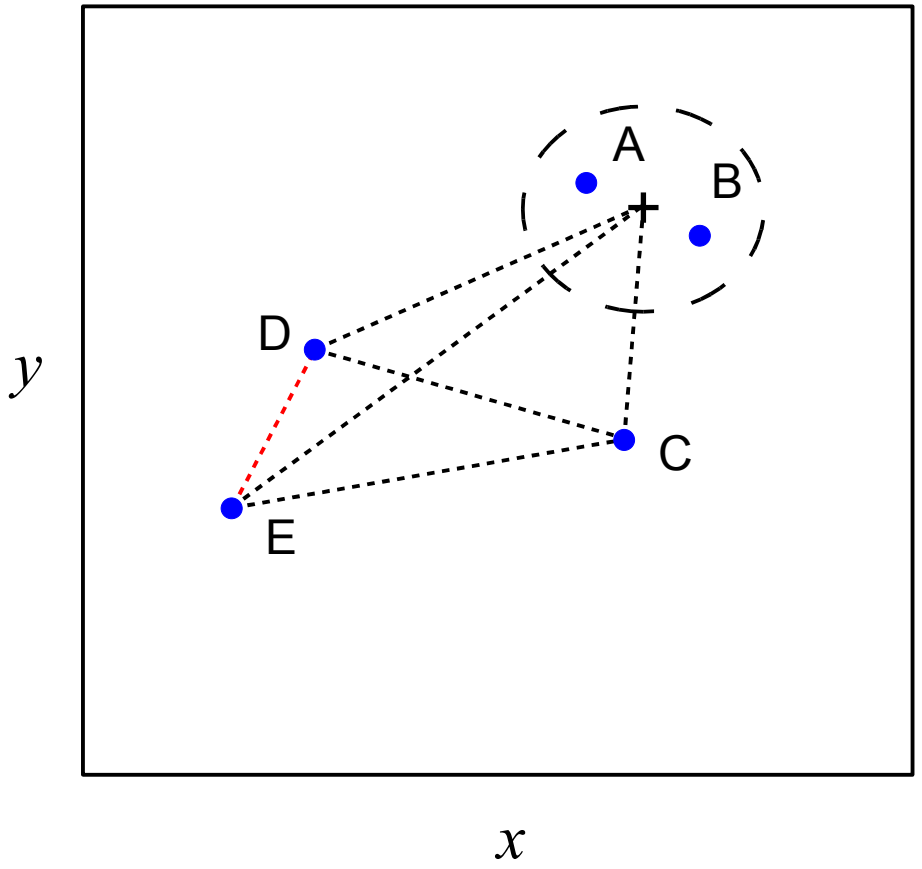


	x	y
A	0.69	0.82
B	0.76	0.77
C	0.71	0.43
D	0.28	0.55
E	0.17	0.32

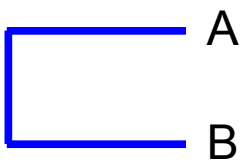


Cluster Analysis

- A geometric method for visualizing the similarity structure of patterns in a multi-dimensional space

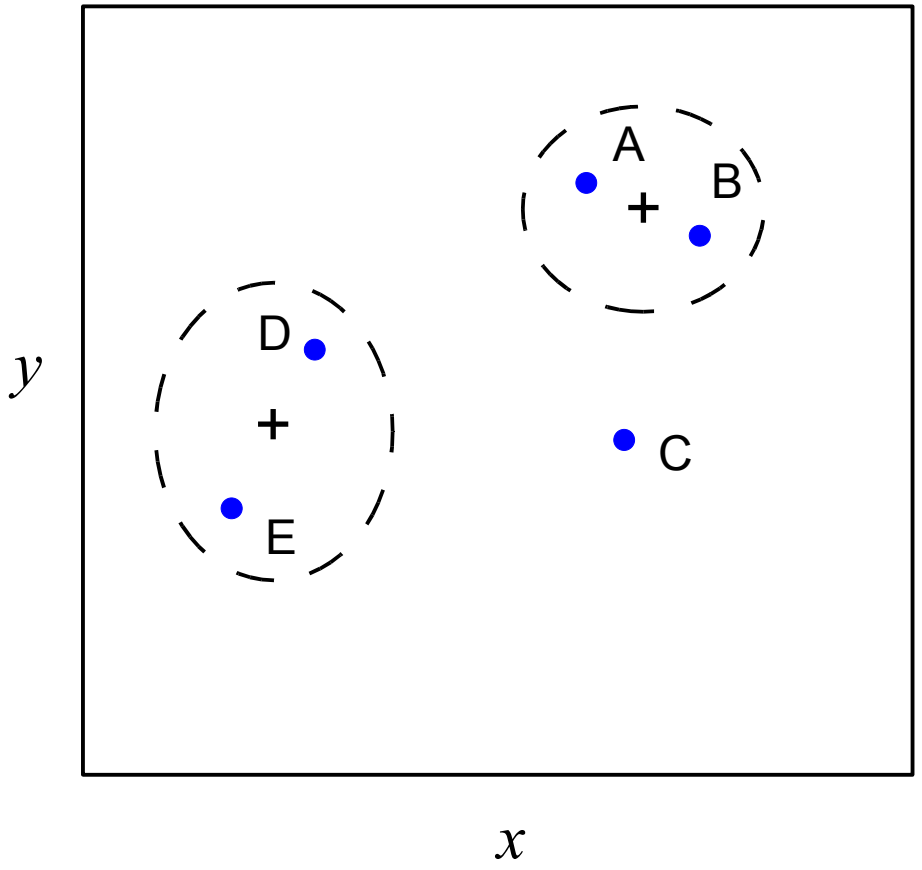


	x	y
A	0.69	0.82
B	0.76	0.77
C	0.71	0.43
D	0.28	0.55
E	0.17	0.32

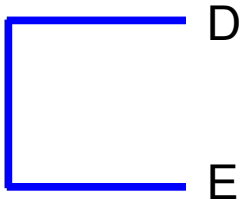


Cluster Analysis

- A geometric method for visualizing the similarity structure of patterns in a multi-dimensional space

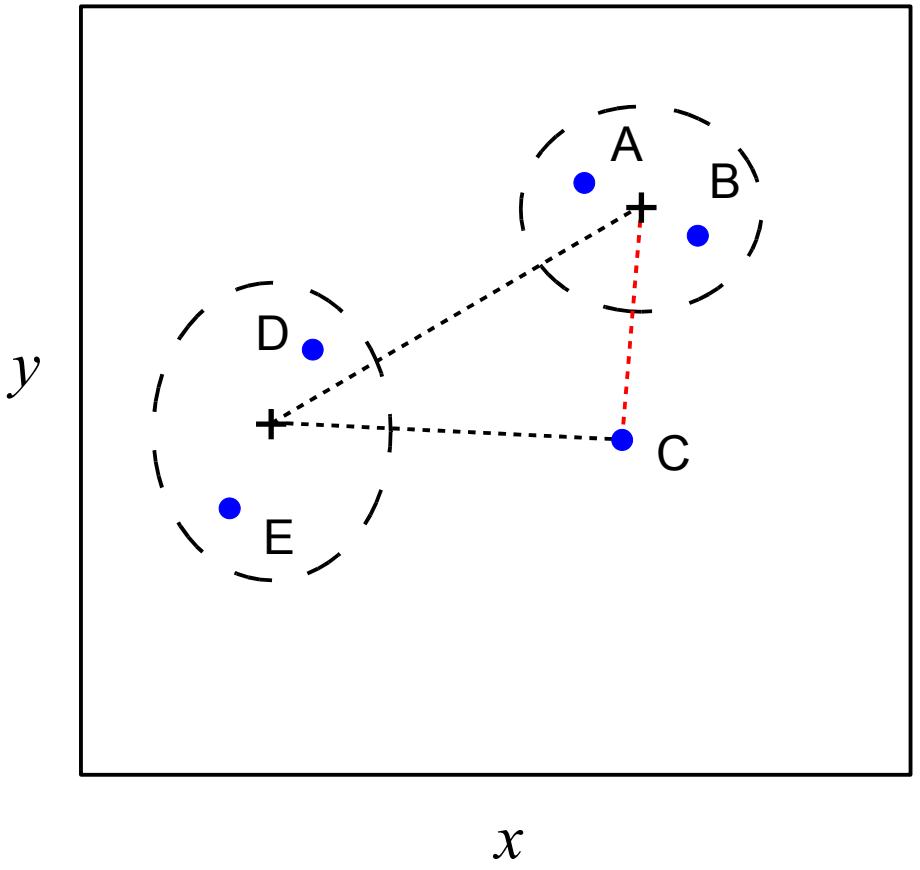


	x	y
A	0.69	0.82
B	0.76	0.77
C	0.71	0.43
D	0.28	0.55
E	0.17	0.32

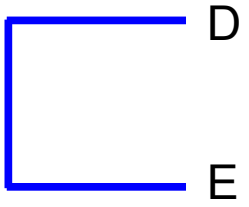


Cluster Analysis

- A geometric method for visualizing the similarity structure of patterns in a multi-dimensional space

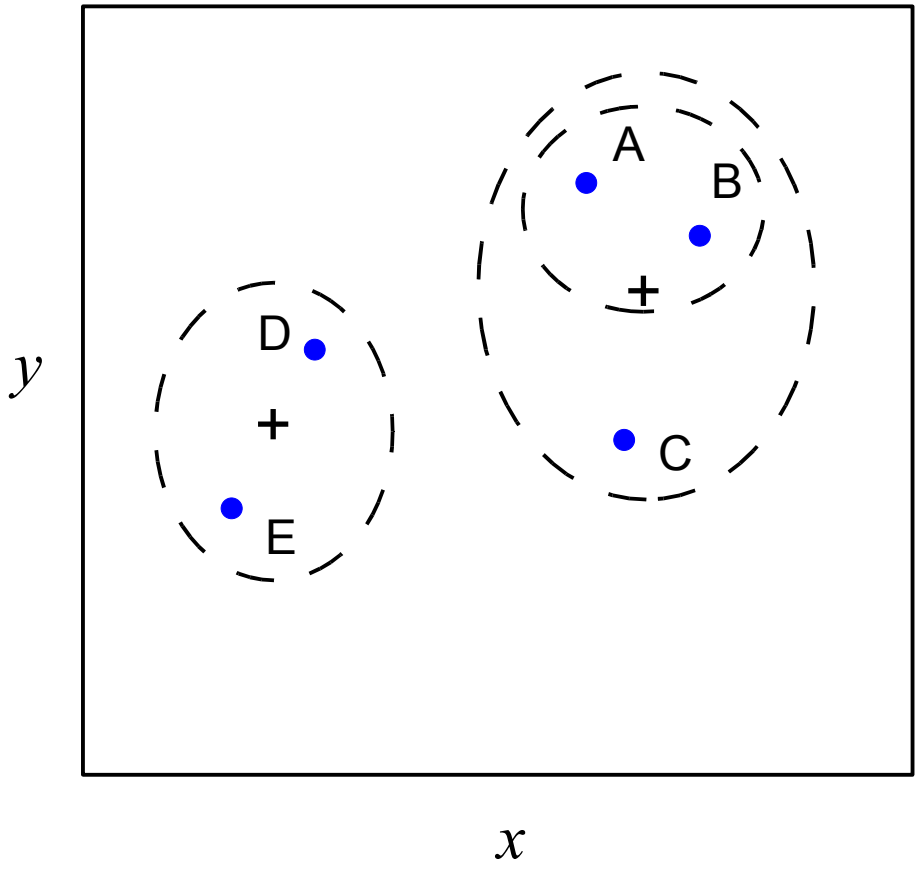


	x	y
A	0.69	0.82
B	0.76	0.77
C	0.71	0.43
D	0.28	0.55
E	0.17	0.32

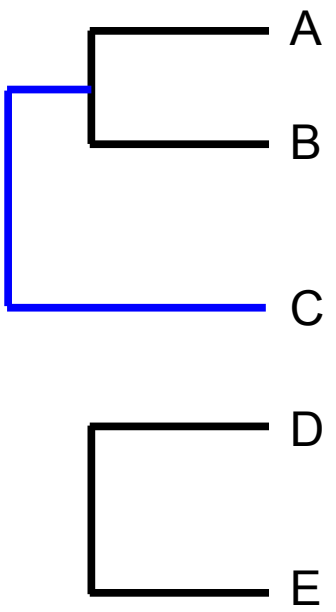


Cluster Analysis

- A geometric method for visualizing the similarity structure of patterns in a multi-dimensional space

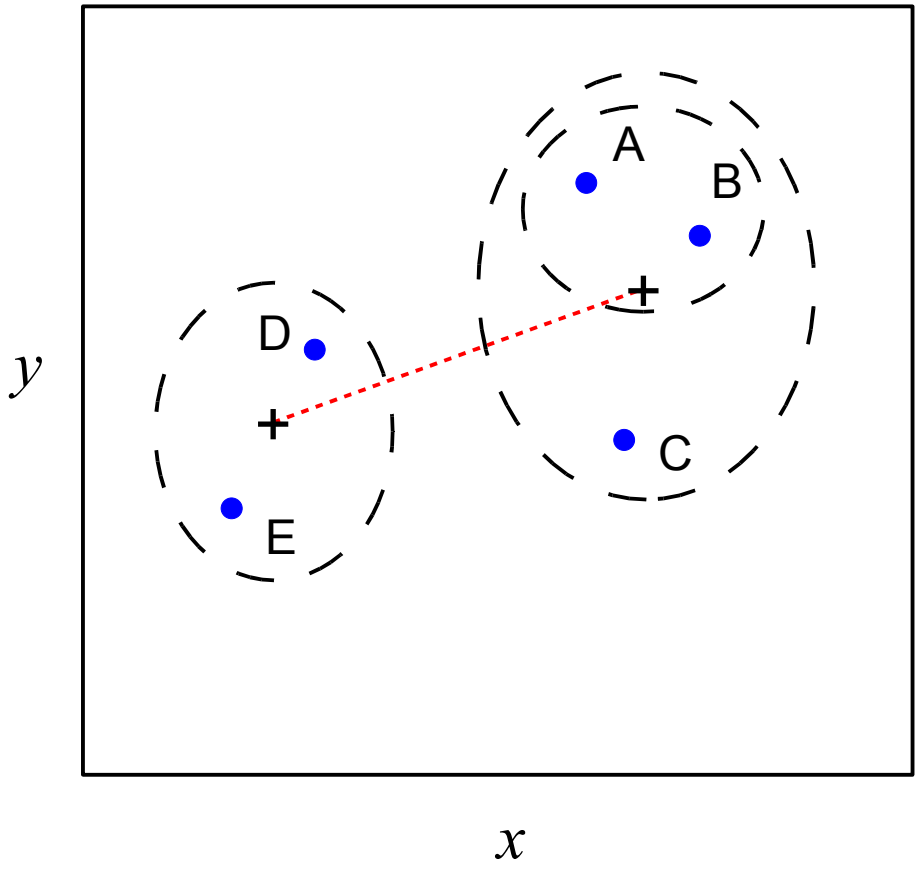


	x	y
A	0.69	0.82
B	0.76	0.77
C	0.71	0.43
D	0.28	0.55
E	0.17	0.32

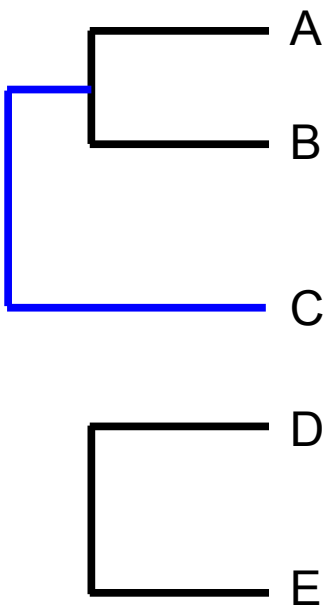


Cluster Analysis

- A geometric method for visualizing the similarity structure of patterns in a multi-dimensional space

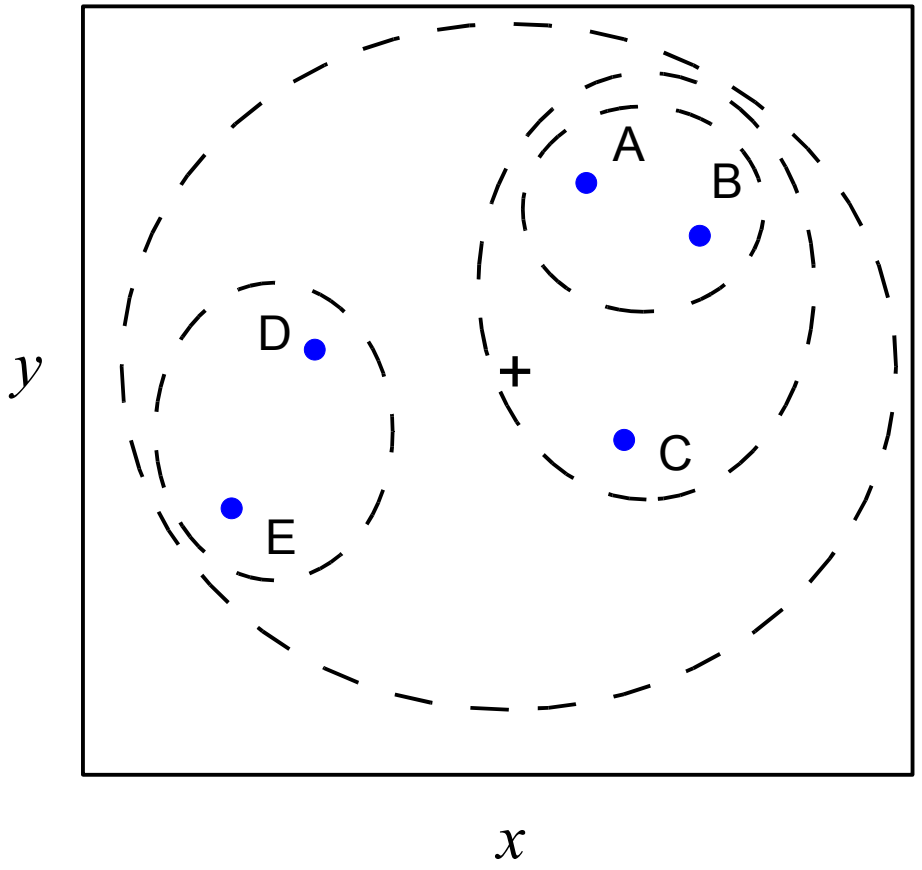


	x	y
A	0.69	0.82
B	0.76	0.77
C	0.71	0.43
D	0.28	0.55
E	0.17	0.32

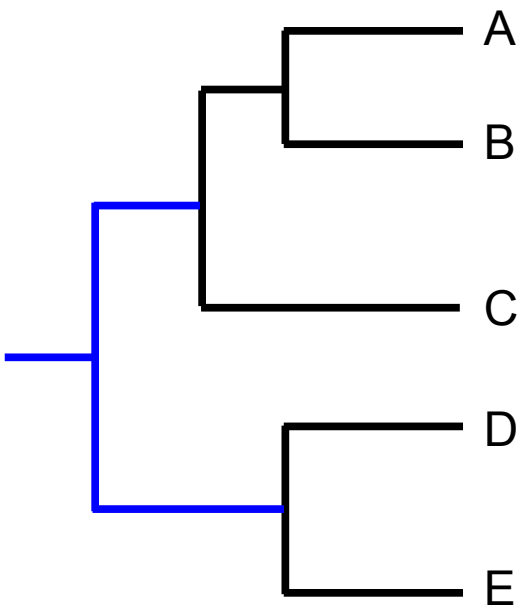


Cluster Analysis

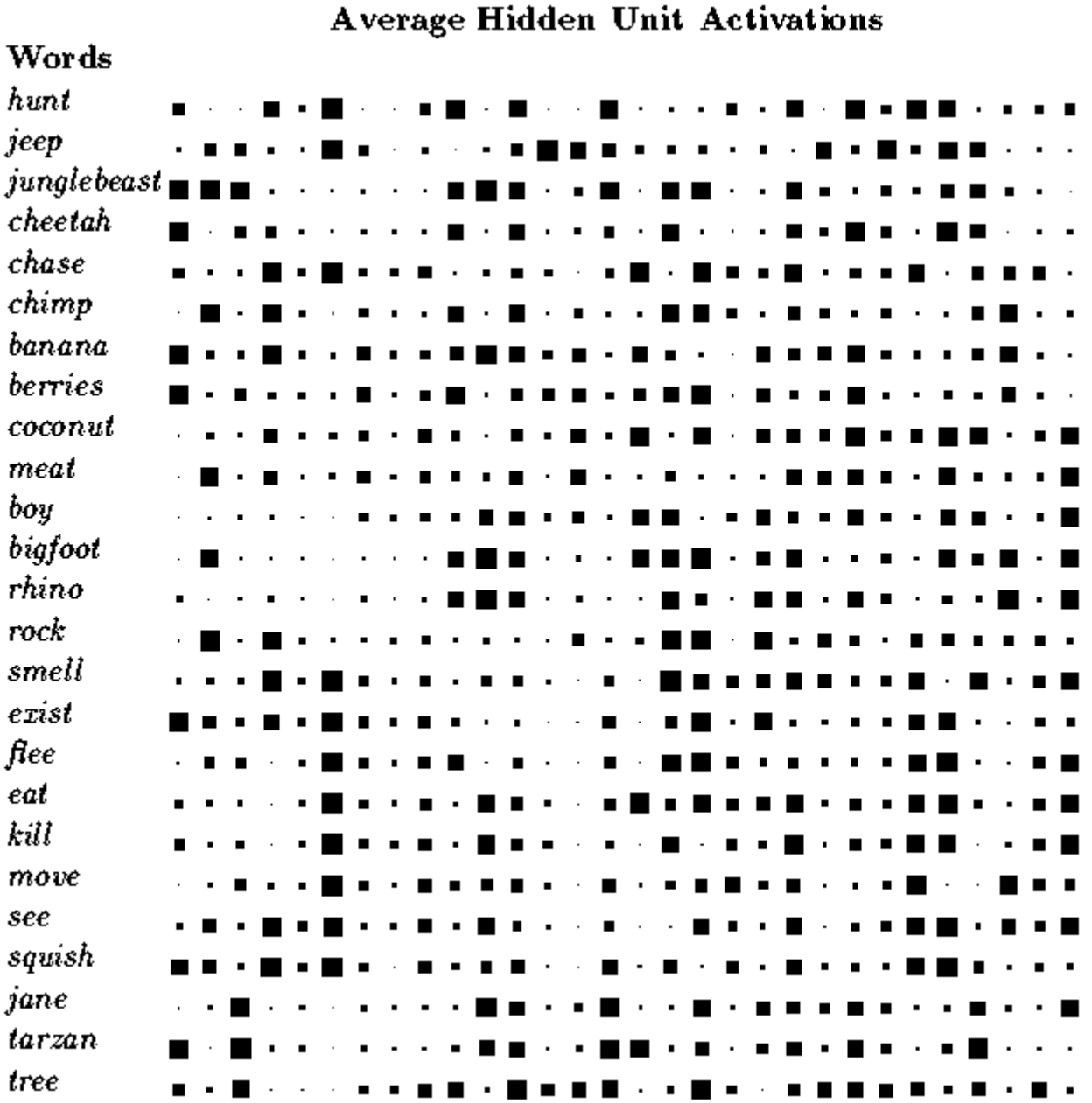
- A geometric method for visualizing the similarity structure of patterns in a multi-dimensional space



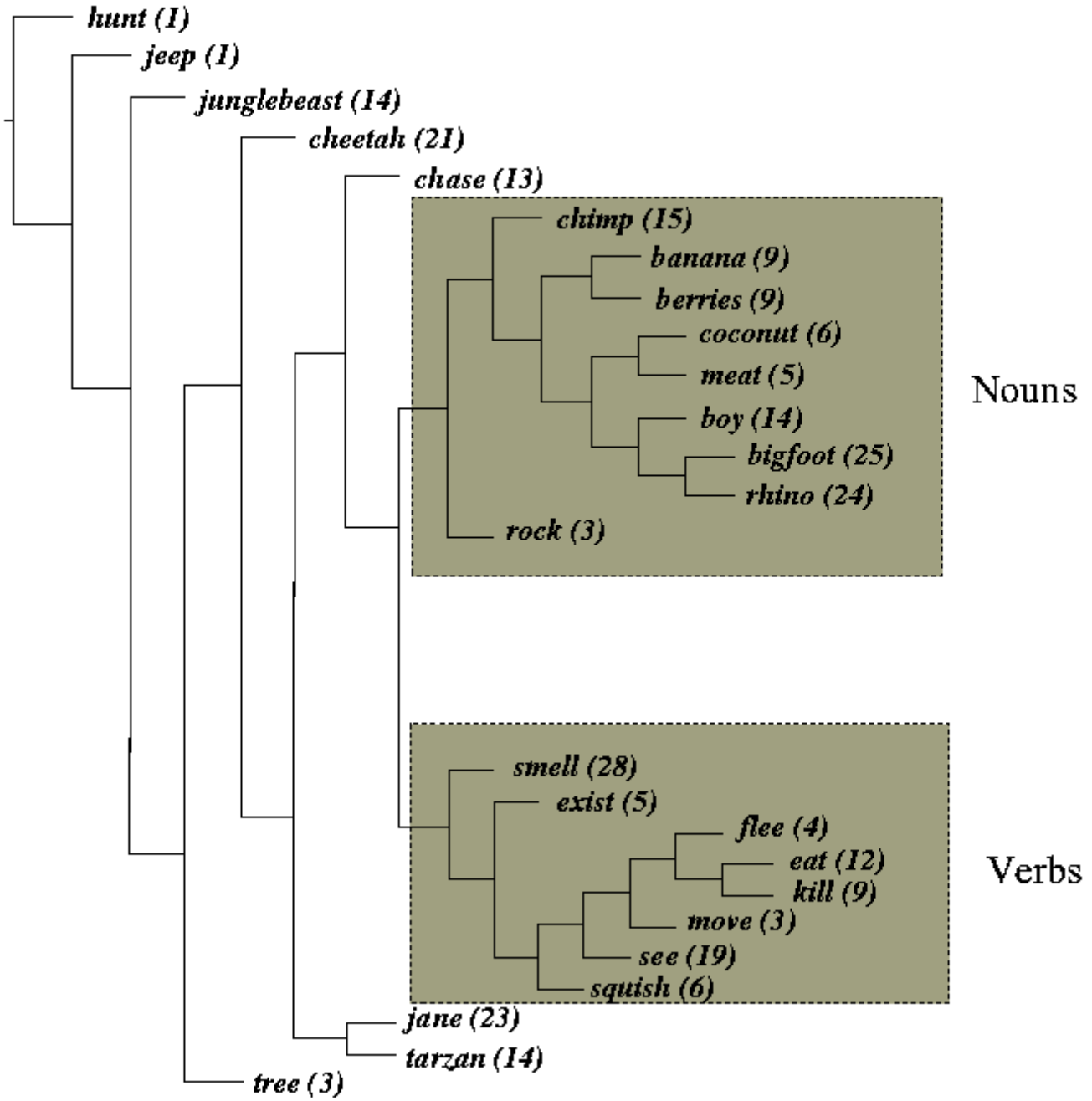
	x	y
A	0.69	0.82
B	0.76	0.77
C	0.71	0.43
D	0.28	0.55
E	0.17	0.32



Cluster Analysis of Word Representations

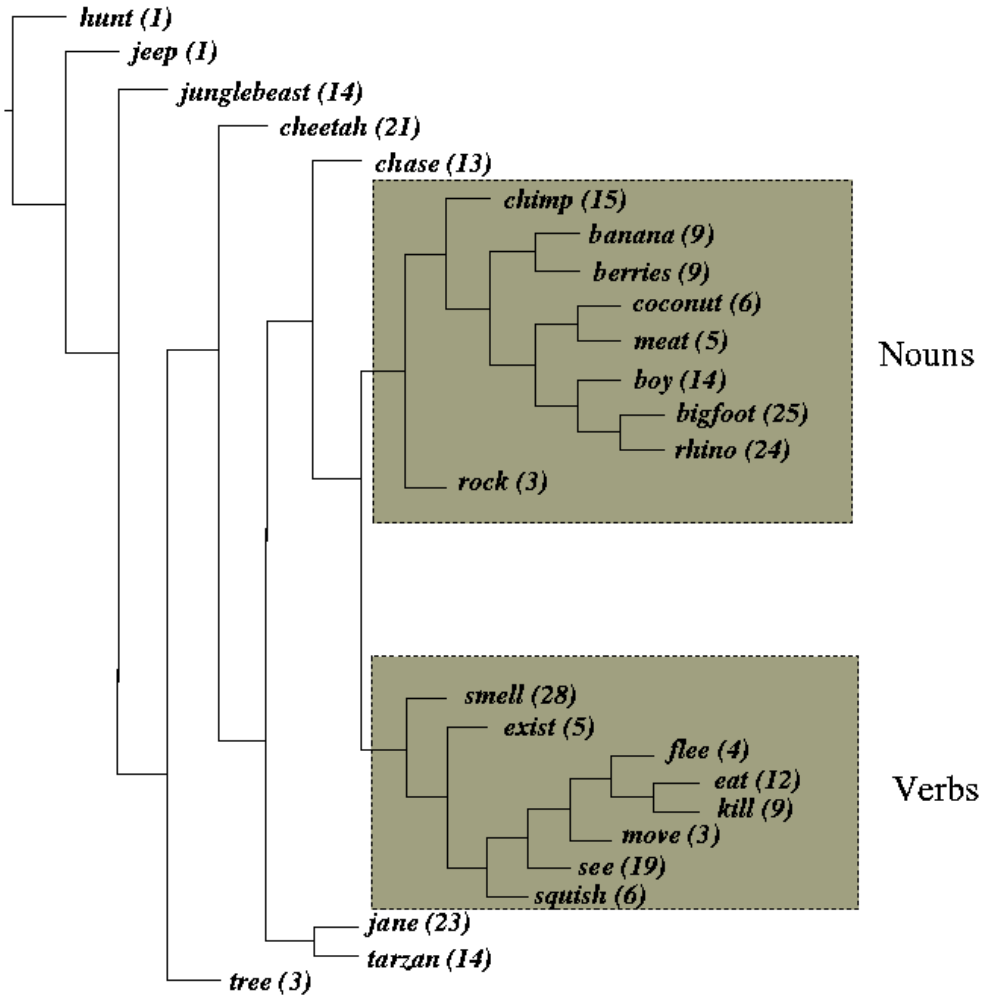


Cluster Analysis of Word Representations



Cluster Analysis of Word Representations

- Good but not perfect separation of nouns/verbs
- Squishable foods (*banana*, *berries*) clustered together
- Non-squishable foods (*meat*, *coconut*) clustered together
- Aggressive nouns (*rhino*, *bigfoot*) clustered together
- Training on 300 sentences instead of 100 reduces the number of exceptions to five
- Word representations reflect the structure of the grammar



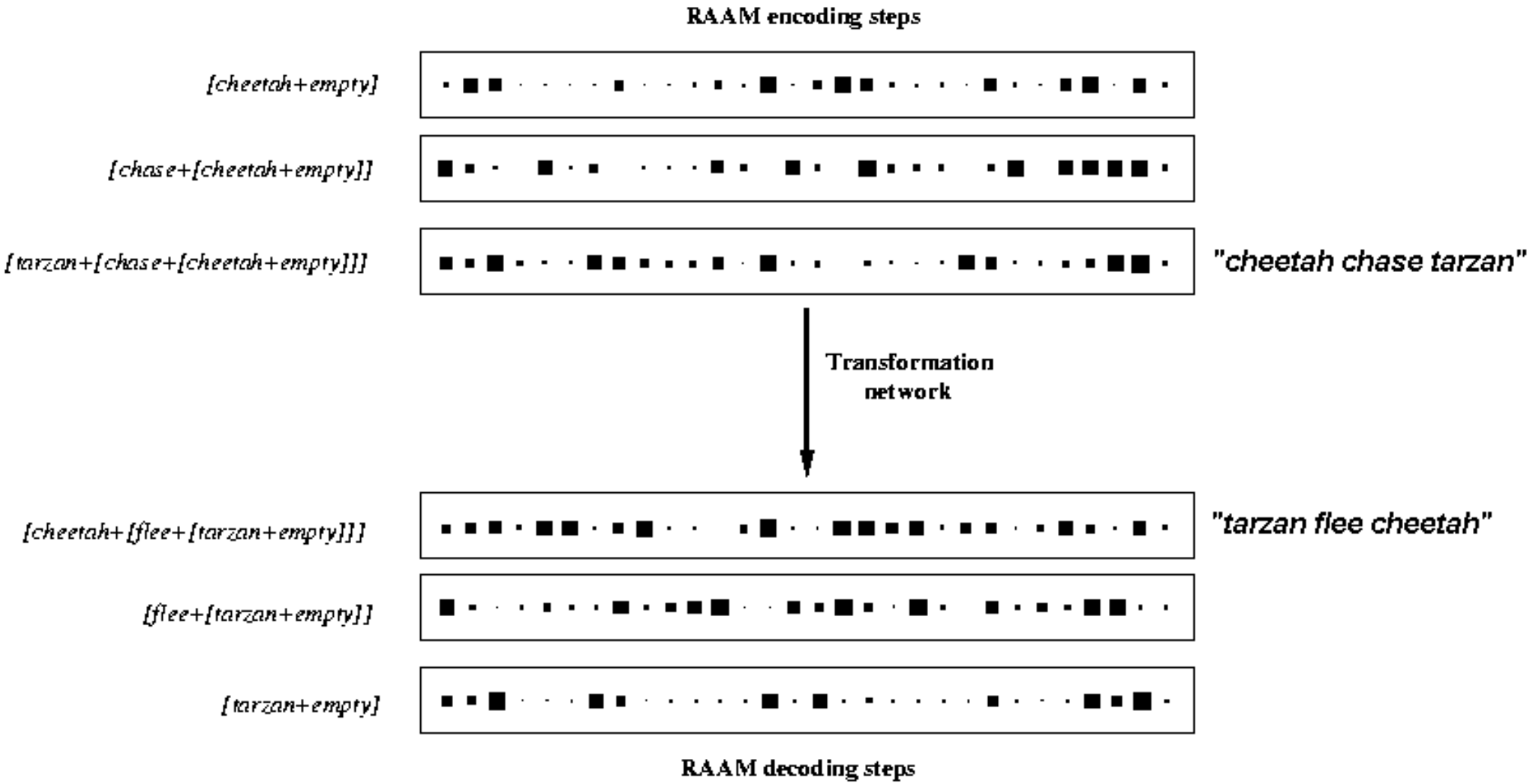
Experiments

- Test 4: Can the network process sentences **holistically**?
- Task: transform "chase" sentences into "flee" sentences

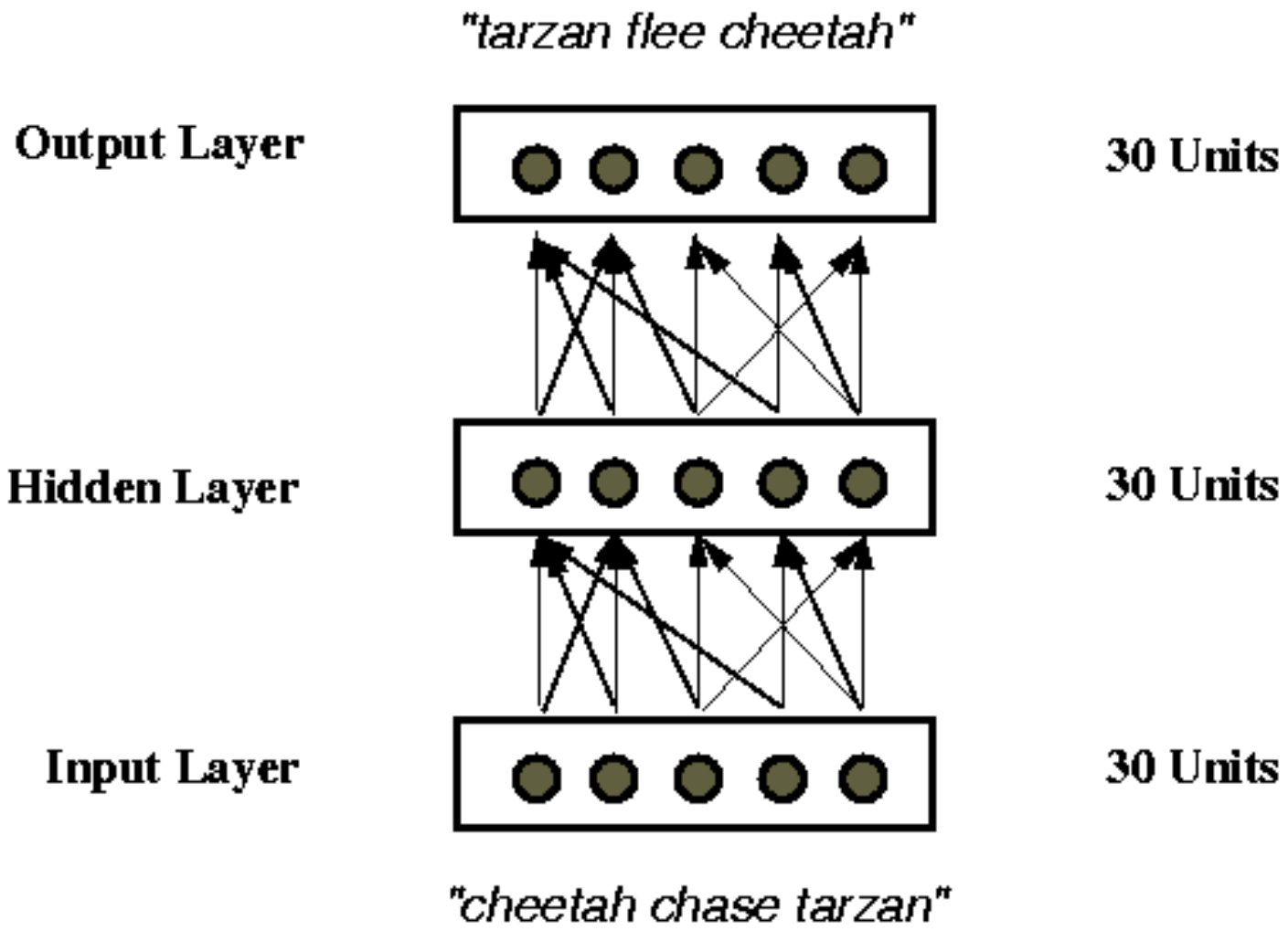
NOUN1 *chase* NOUN2 ⇒ NOUN2 *flee* NOUN1

- Use a separate neural network to learn this transformation
  - ordinary backpropagation network (not a RAAM)
  - operates directly on **RAAM-encoded** sentences
  - does **not** decode sentences into their constituent parts
  - input: a pattern that encodes a "chase" sentence
  - output: a pattern that encodes the transformed sentence

Example: *"cheetah chase tarzan"*



Transformation Network



Experiments

- Test 4: Can the network process sentences **holistically**?
- Step 1:
  - Generated a new training corpus for the RAAM
    - 20 *chase*/*flee* sentence pairs
    - 110 other valid sentences
  - Trained RAAM as before for ~ 3700 training cycles
  - Encoded the 20 *chase*/*flee* sentence pairs, plus 4 novel *chase*/*flee* sentence pairs, using the trained RAAM

Experiments

- Test 4: Can the network process sentences **holistically**?
- Step 2:
  - Trained transformation network on 16 of the 24 *chase*/*flee* sentence pairs for ~ 75 training cycles
  - Tested transformation network on the remaining 8 *chase*/*flee* sentence pairs
    - 4 of these pairs had been used to train the RAAM
    - 4 of these pairs were completely novel
- Results:
  - 100% correct on RAAM-trained sentence pairs (4 out of 4)
  - 75% correct on novel sentence pairs (3 out of 4)
  - Error: *junglebeast chase chimp* ⇒ *chimp flee cheetah*

Summary

- RAAM networks **can** represent compositional linguistic structures such as words, sentences, and parse trees
- These representations can be **highly distributed**
- The similarity structure of these representations reflects the way words are used in context, even though **no explicit knowledge** of the grammar was provided to the network
- Neural networks can be trained to process representations **holistically**, without having to first break them up into their constituent parts
- This is **fundamentally different** from the way in which symbolic AI programs process linguistic structures
- Fodor and Pylyshyn may have seriously **underestimated** the potential of connectionism for linguistic processing