DL for NLP
From the Basics to Research

Richard Socher

Research work joint with the MetaMind-Salesforce team
Caiming Xiong, Stephen Merity, James Bradbury,
Ankit Kumar, Ozan Irsoy and others

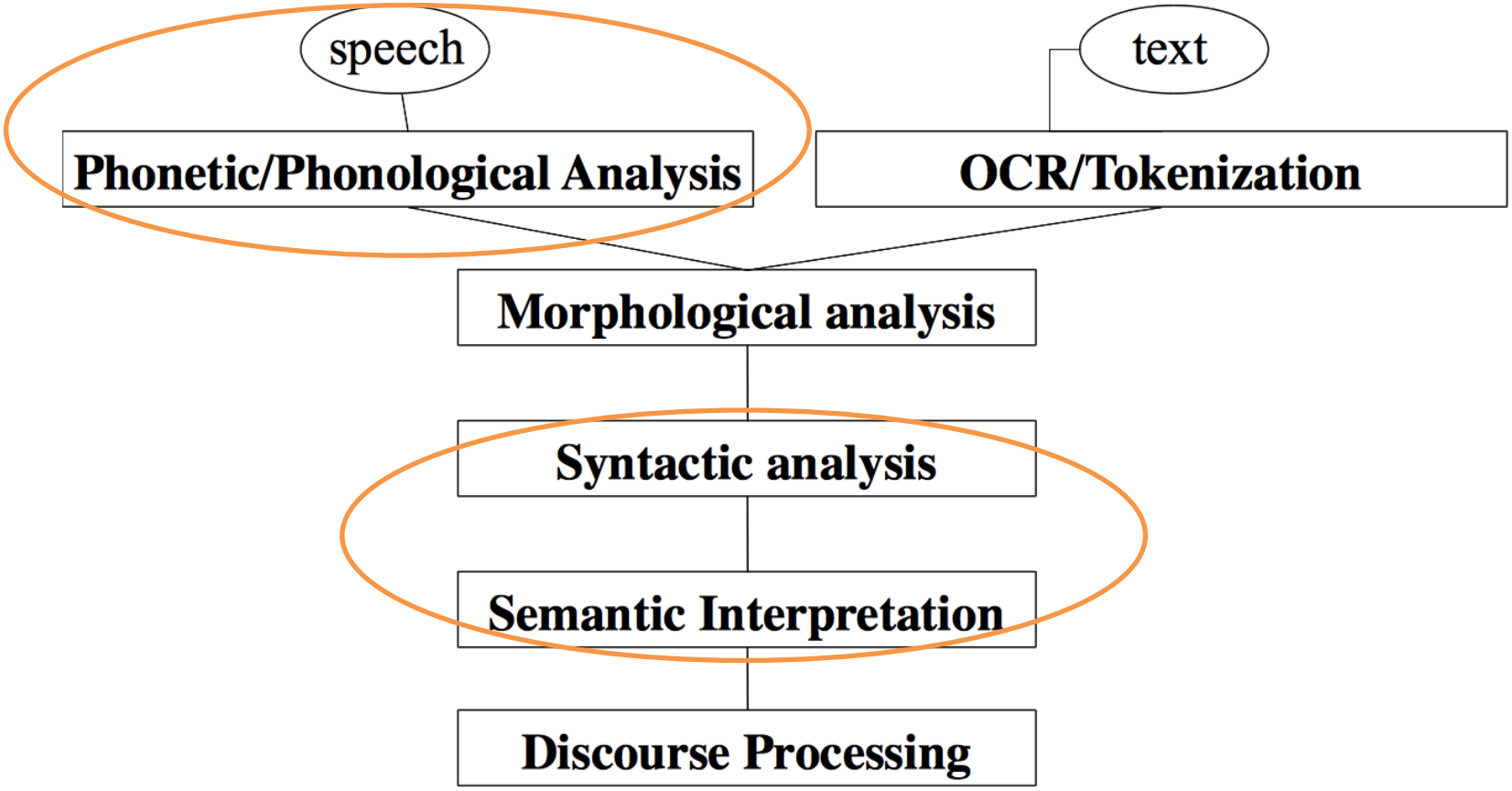
What is Natural Language Processing?

- Natural language processing is a field at the intersection of
  - computer science
  - artificial intelligence
  - and linguistics.
- Goal: for computers to process or "understand" natural language in order to perform tasks that are useful, e.g.
  - Question Answering

- Fully **understanding and representing** the **meaning** of language (or even defining it) is an illusive goal.

- Perfect language understanding is AI-complete

NLP Levels


speech
text
Phonetic/Phonological Analysis
OCR/Tokenization
Morphological analysis
Syntactic analysis
Semantic Interpretation
Discourse Processing

Why is NLP hard?

- Complexity in representing, learning and using linguistic/situational/world/visual knowledge
- Jane hit June and then **she** [fell/ran].
- Ambiguity: “I made her duck”

(A tiny sample of) NLP Applications

- Applications range from simple to complex:

- Spell checking, keyword search, finding synonyms

- Extracting information from websites such as
  - product price, dates, location, people or company names
- Classifying, reading level of school texts, positive/negative sentiment of longer documents

- Machine translation
- Question answering or automated email replies
- Spoken dialog systems

Representations for Language Tasks: Machine Translation

- Many levels of translation have been tried in the past:
- Traditional MT systems are very large complex systems

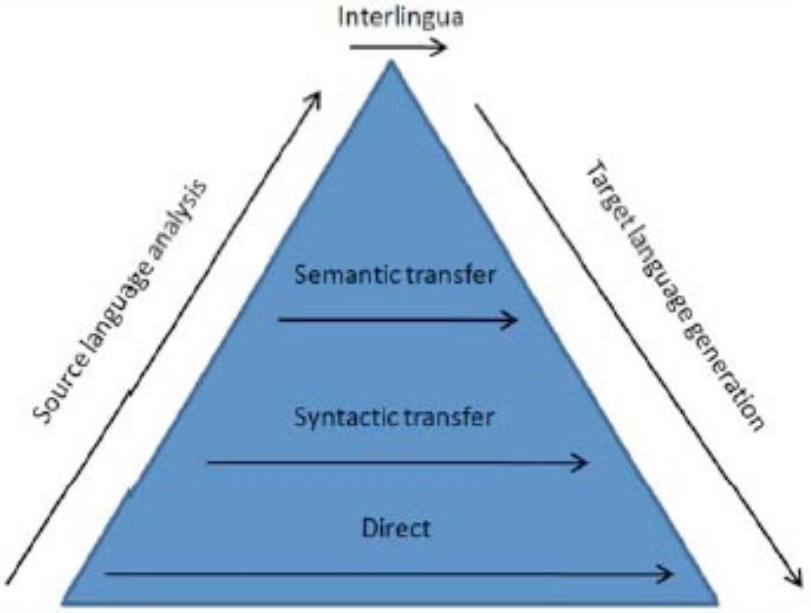
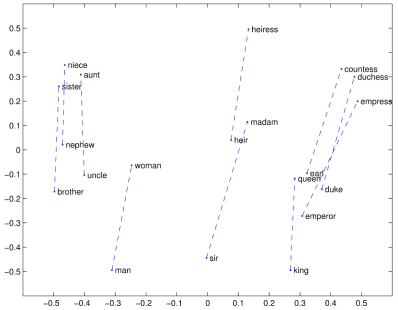


Figure 1: The Vauquois triangle

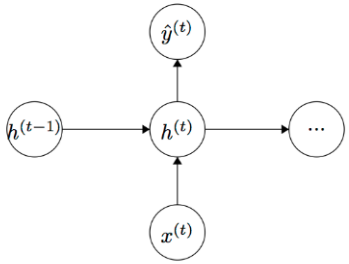
- What do you think is the interlingua for the DL approach to translation?
- Representation for all levels: Vectors!

Outline

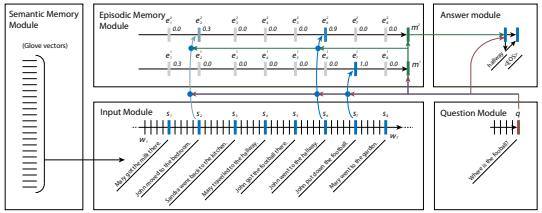
1. Words
   Basics: Word2vec and Glove



2. Sentences (~)
   Basics: Recurrent neural networks



3. Multiple sentences
   Research:
   Dynamic memory networks



How to represent meaning?

Common answer: Use a taxonomy like WordNet that has hypernyms (is-a) relationships and synonym sets (good):

```
from nltk.corpus import wordnet as wn
panda = wn.synset('panda.n.01')
hyper = lambda s: s.hypernyms()
list(panda.closure(hyper))
```

```
[Synset('procyonid.n.01'),
Synset('carnivore.n.01'),
Synset('placental.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')]
```

```
S: (adj) full, good
S: (adj) estimable, good, honorable, respectable
S: (adj) beneficial, good
S: (adj) good, just, upright
S: (adj) adept, expert, good, practiced,
proficient, skillful
S: (adj) dear, good, near
S: (adj) good, right, ripe
…
S: (adv) well, good
S: (adv) thoroughly, soundly, good
S: (n) good, goodness
S: (n) commodity, trade good, good
```

Problems with discrete representations

- Great as resource but missing nuances, e.g. **synonyms**: adept, expert, good, practiced, proficient, skillful?

- Missing new words (impossible to keep up to date): wicked, badass, nifty, crack, ace, wizard, genius, ninjia
- Subjective
- Requires human labor to create and adapt
- Hard to compute accurate word similarity

Instead: Use distributional similarity

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 statistical 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* ↗

Window based cooccurence matrix

Simple example: For each word define:

- Window length 1 (more common: 5 - 10)
- Symmetric window around each word
- Example corpus:
  - I like deep learning.
  - I like NLP.
  - I enjoy flying.

Window based cooccurence matrix

- Example corpus:
  - I like deep learning.
    I like NLP.
    I enjoy flying.

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

- Could run SVD on this matrix

word2vec (Mikolov et al 2013)

- Instead of capturing cooccurrence counts directly
- Predict surrounding words of every word

- Faster and can easily incorporate a new sentence/document or add a word to the vocabulary

Details of Word2Vec

- Predict surrounding words in a window of length m of every word.
- Objective function: Maximize the log probability of any context word given the current center word:
- $$J(\theta) = \frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leq j\leq m, j\neq 0} \log p(w_{t+j}|w_t)$$
- Where represents all variables we optimize

Details of Word2Vec

- Predict surrounding words in a window of length m of every word
- For $p(w_{t+j}|w_t)$ the simplest first formulation is

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

- where o is the outside (or output) word id, c is the center word id, v and u are “center” and “outside” vectors of indices c and o
- Every word has two vectors!
- This is essentially “dynamic” logistic regression
- For improved versions + detailed derivation, see cs224d.stanford.edu

Count based vs direct prediction

LSA, HAL (Lund & Burgess),
COALS (Rohde et al),
Hellinger-PCA (Lebret & Collobert)

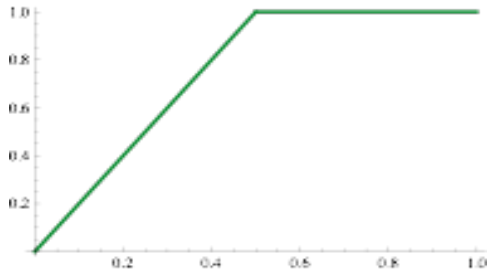
- Fast training
- Efficient usage of statistics
- Primarily used to capture word similarity
- Disproportionate importance given to large counts

- NNLM, HLBL, RNN, Skip-gram/CBOW, (Bengio et al; Collobert & Weston; Huang et al; Mnih & Hinton; Mikolov et al; Mnih & Kavukcuoglu)

- Scales with corpus size
- Inefficient usage of statistics
- Generate improved performance on other tasks
- Can capture complex patterns beyond word similarity

Combining the best of both worlds: GloVe (Pennington et al. 2014)

$$J(\theta) = \frac{1}{2} \sum_{i,j=1}^{W} f(P_{ij})(u_i^T v_j - \log P_{ij})^2$$

$f \sim$ 

- Fast training
- Scalable to huge corpora
- Good performance even with small corpus, and small vectors

Glove results

Nearest words to frog:

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

litoria

leptodactylidae

rana

eleutherodactylus

Intrinsic word vector evaluation

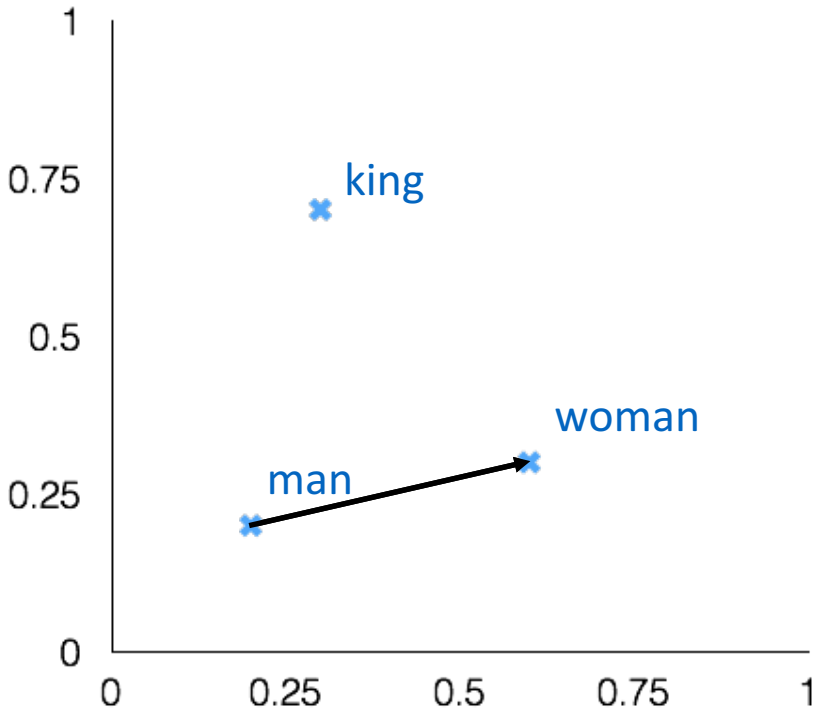
- Word Vector Analogies

a:b :: c:?

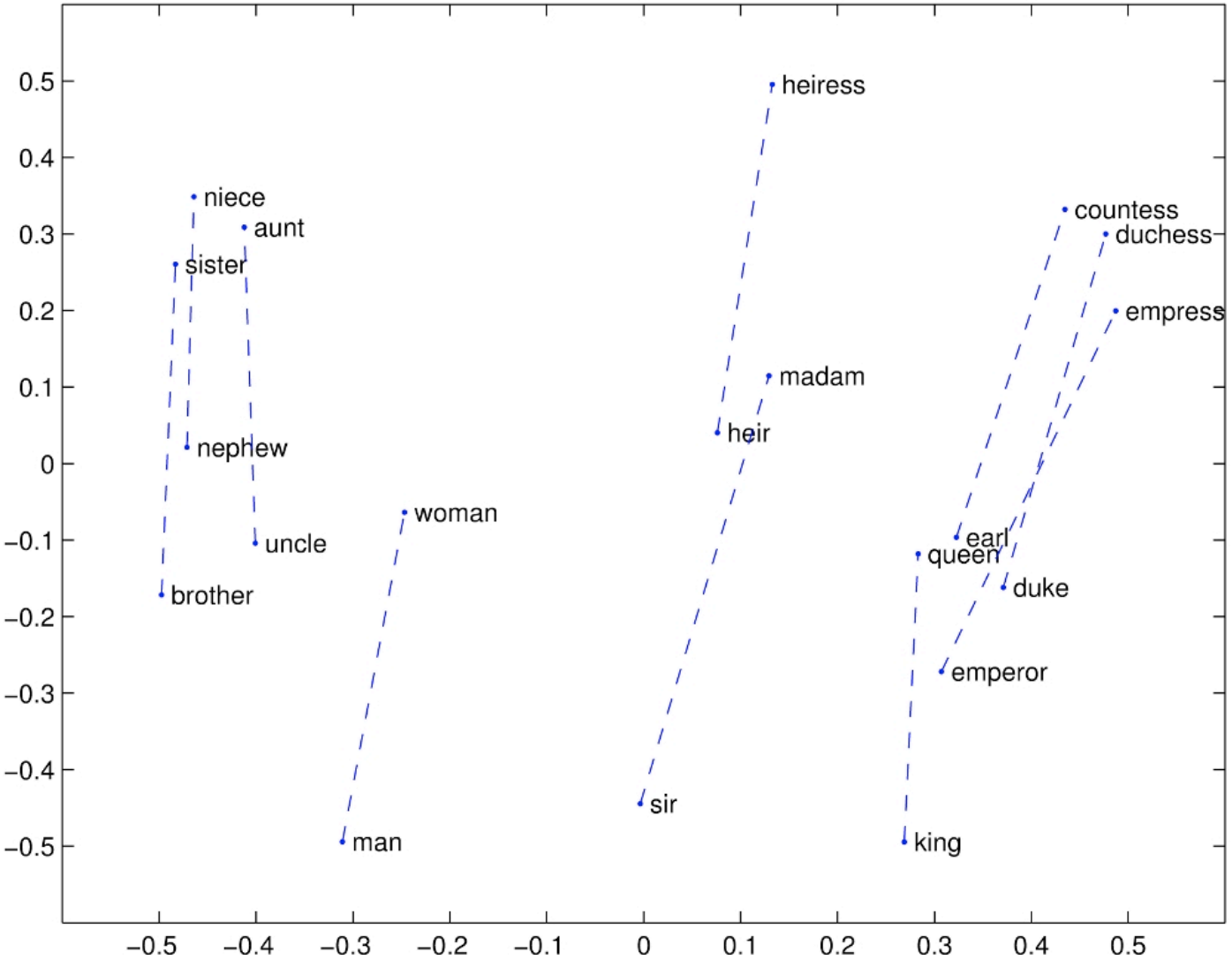
man:woman :: king:?

$$d = \arg\max_i \frac{(x_b - x_a + x_c)^T x_i}{||x_b - x_a + x_c||}$$

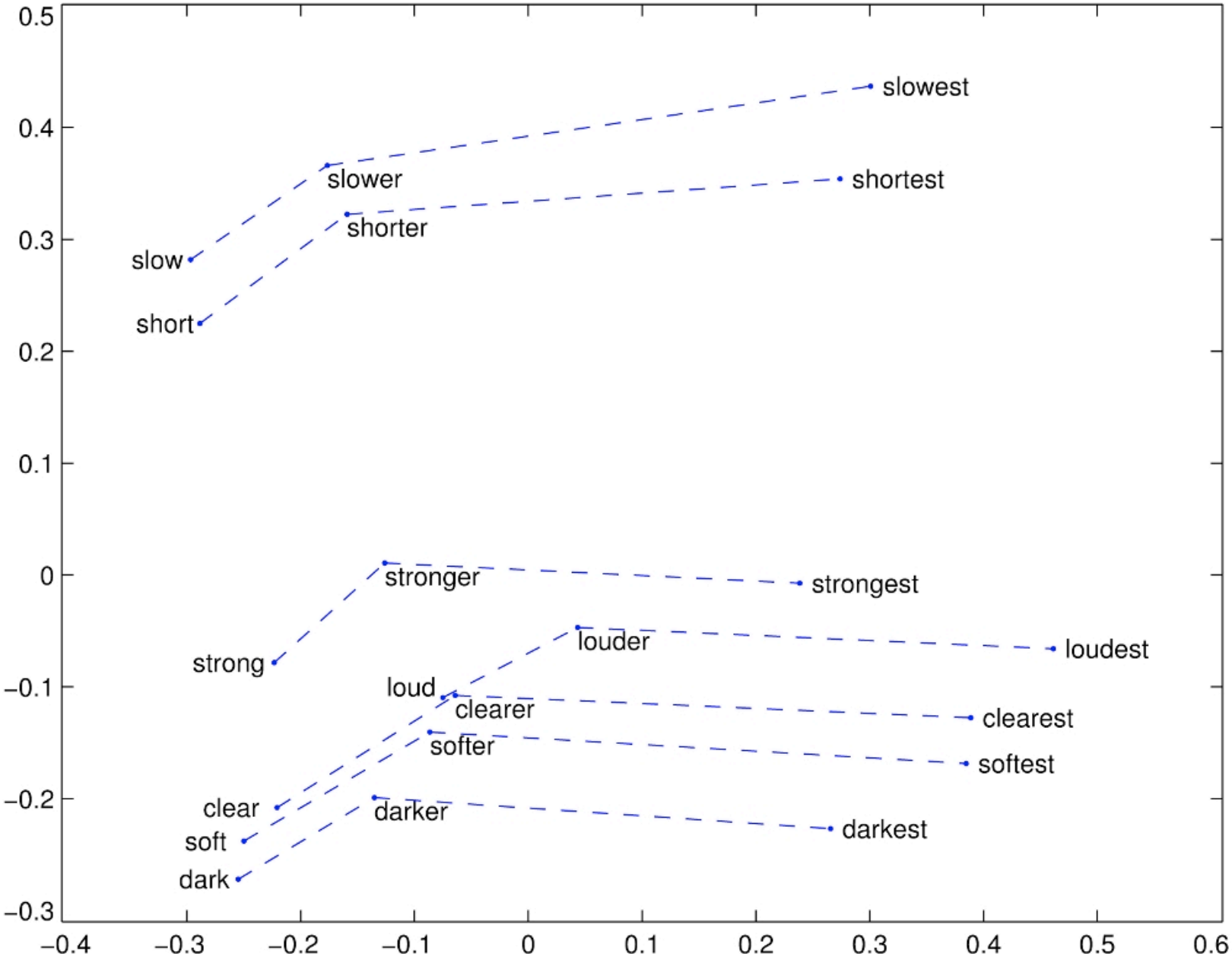
- Evaluate word vectors by how well their cosine distance after addition captures intuitive semantic and syntactic analogy questions
- Discarding the input words from the search!
- Problem: What if the information is there but not linear?



Glove Visualizations



Glove Visualizations: Superlatives



Analogy evaluation and hyperparameters

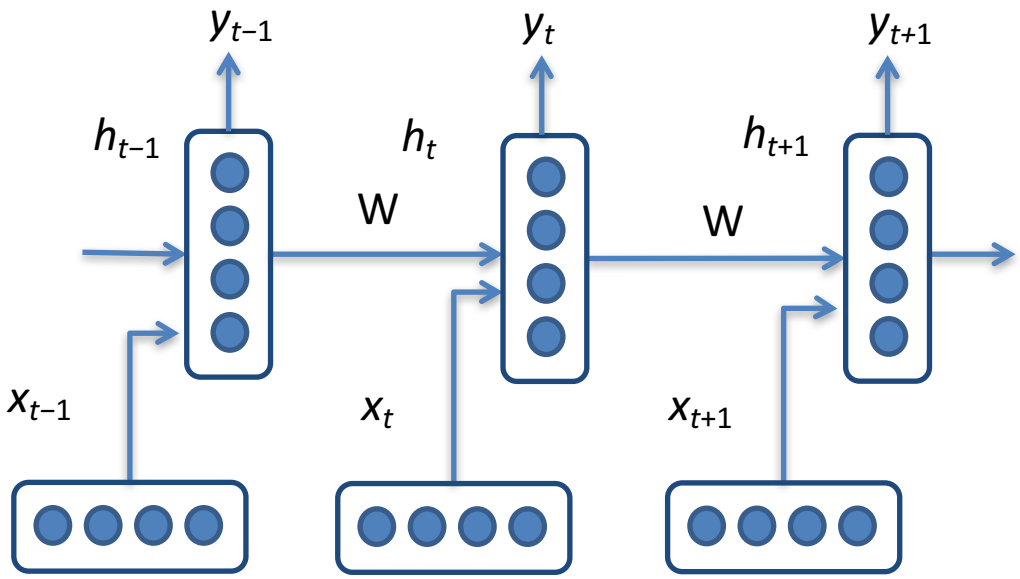
- More data is better

Model	Dim.	Size	Sem.	Syn.	Tot.
ivLBL	100	1.5B	55.9	50.1	53.2
HPCA	100	1.6B	4.2	16.4	10.8
GloVe	100	1.6B	67.5	54.3	60.3
SG	300	1B	61	61	61
CBOW	300	1.6B	16.1	52.6	36.1
vLBL	300	1.5B	54.2	64.8	60.0
ivLBL	300	1.5B	65.2	63.0	64.0
GloVe	300	1.6B	80.8	61.5	70.3
SVD	300	6B	6.3	8.1	7.3
SVD-S	300	6B	36.7	46.6	42.1
SVD-L	300	6B	56.6	63.0	60.1
CBOW†	300	6B	63.6	67.4	65.7
SG†	300	6B	73.0	66.0	69.1
GloVe	300	6B	77.4	67.0	71.7
CBOW	1000	6B	57.3	68.9	63.7
SG	1000	6B	66.1	65.1	65.6
SVD-L	300	42B	38.4	58.2	49.2
GloVe	300	42B	**81.9**	**69.3**	**75.0**

Section 2:
Recurrent neural networks

Recurrent Neural Networks (!)

- Similar to normal neural networks
- RNNs tie the weights at each time step
- Condition the neural network on all previous words



RNN language model

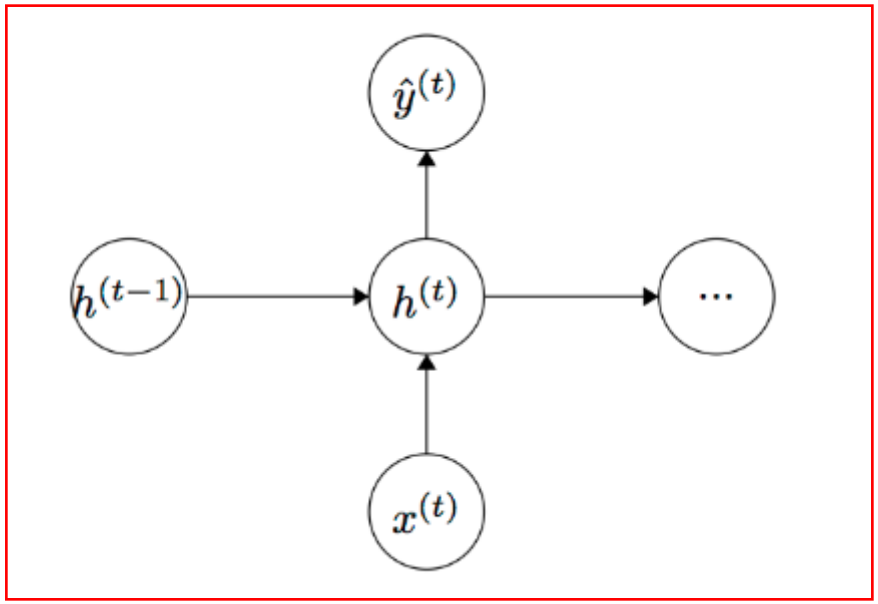
Given list of word **vectors**: $x_1, \ldots, x_{t-1}, x_t, x_{t+1}, \ldots, x_T$

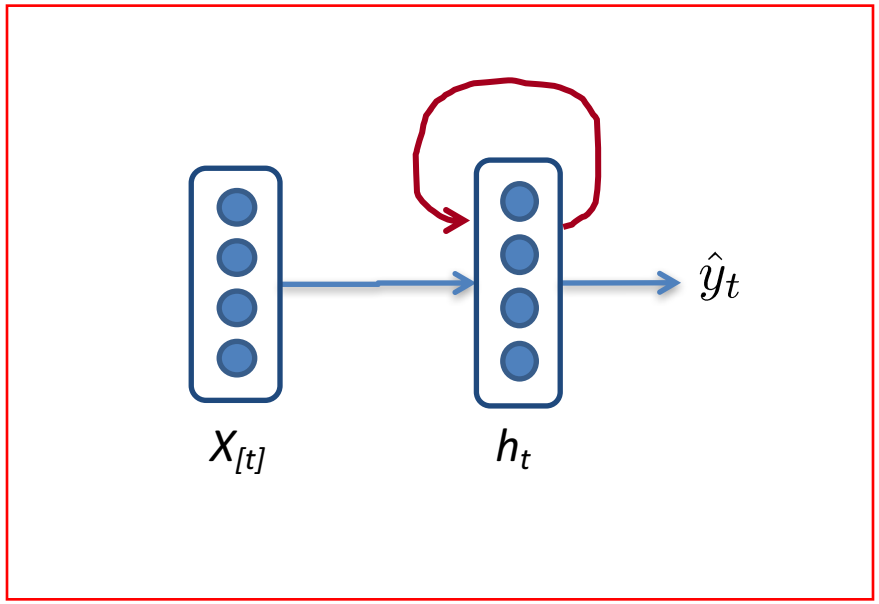
At each time step,
predict the next word:

$$h_t = \sigma\left(W^{(hh)} h_{t-1} + W^{(hx)} x_{[t]}\right)$$

$$\hat{y}_t = \text{softmax}\left(W^{(S)} h_t\right)$$

$$\hat{P}(x_{t+1} = v_j \mid x_t, \ldots, x_1) = \hat{y}_{t,j}$$





RNN language model

Same set of W weights at all time steps!

Everything else is the same:

$$
\begin{aligned}
h_t &= \sigma\left(W^{(hh)} h_{t-1} + W^{(hx)} x_{[t]}\right) \\
\hat{y}_t &= \text{softmax}\left(W^{(S)} h_t\right) \\
\hat{P}(x_{t+1} = v_j \mid x_t, \ldots, x_1) &= \hat{y}_{t,j}
\end{aligned}
$$

$h_0 \in \mathbb{R}^{D_h}$ is initialization vector for hidden layer at first step

The next word is the "class." Performance measured in

Perplexity: 2^J where: $J = -\frac{1}{T} \sum_{t=1}^{T} \sum_{j=1}^{|V|} y_{t,j} \log \hat{y}_{t,j}$

Main RNN improvement: Better Units

- More complex hidden unit computation in recurrence!
- Gated Recurrent Units (GRU) introduced by Cho et al. 2014. Special case of an LSTM Hochreiter and Schmidhuber 1997
- Main ideas:
  - keep around memories to capture long distance dependencies
  - allow error messages to flow at different strengths depending on the inputs

GRUs

- Standard RNN computes hidden layer at next time step directly: $h_t = f\left(W^{(hh)}h_{t-1} + W^{(hx)}x_t\right)$
- GRU first computes an update **gate** (another layer) based on current input word vector and hidden state

$$z_t = \sigma\left(W^{(z)}x_t + U^{(z)}h_{t-1}\right)$$

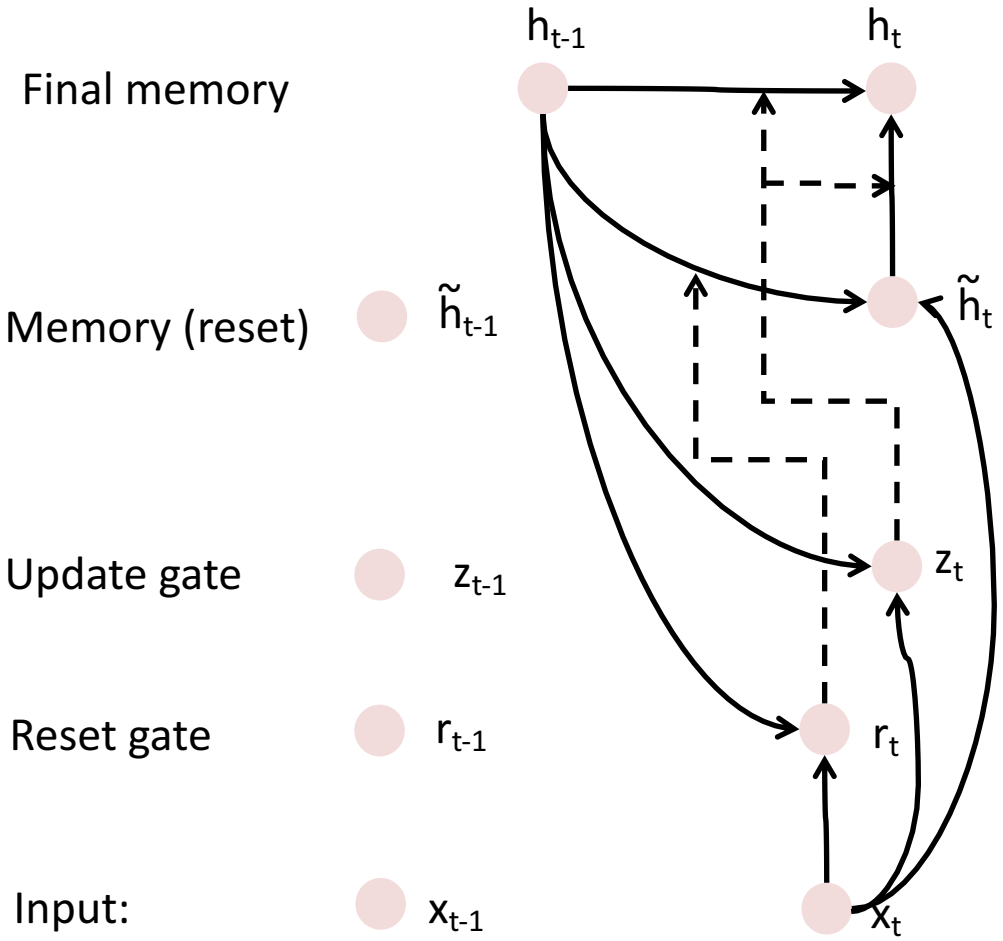
- Compute reset gate similarly but with different weights

$$r_t = \sigma\left(W^{(r)}x_t + U^{(r)}h_{t-1}\right)$$

GRUs

- Update gate $z_t = \sigma\left(W^{(z)}x_t + U^{(z)}h_{t-1}\right)$
- Reset gate $r_t = \sigma\left(W^{(r)}x_t + U^{(r)}h_{t-1}\right)$
- New memory content: $\tilde{h}_t = \tanh\left(Wx_t + r_t \circ Uh_{t-1}\right)$
  If reset gate unit is ~0, then this ignores previous memory and only stores the new word information
- Final memory at time step combines current and previous time steps: $h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h}_t$

Attempt at a clean illustration



$$z_t = \sigma\left(W^{(z)}x_t + U^{(z)}h_{t-1}\right)$$

$$r_t = \sigma\left(W^{(r)}x_t + U^{(r)}h_{t-1}\right)$$

$$\tilde{h}_t = \tanh\left(Wx_t + r_t \circ Uh_{t-1}\right)$$

$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h}_t$$

GRU intuition

- If reset is close to 0, ignore previous hidden state → Allows model to drop information that is irrelevant in the future

$$z_t = \sigma\left(W^{(z)}x_t + U^{(z)}h_{t-1}\right)$$
$$r_t = \sigma\left(W^{(r)}x_t + U^{(r)}h_{t-1}\right)$$
$$\tilde{h}_t = \tanh\left(Wx_t + r_t \circ Uh_{t-1}\right)$$
$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h}_t$$

- Update gate z controls how much of past state should matter now.
  - If z close to 1, then we can copy information in that unit through many time steps! **Less vanishing gradient**!
- Units with short-term dependencies often have reset gates very active

More amazing RNN work

- In Quoc's lecture

Basic lego blocks

- Word vectors and RNNs are the two most important concepts for deepNLP
- Congrats!
- Now we can play with these lego blocks

Problem with all models so far

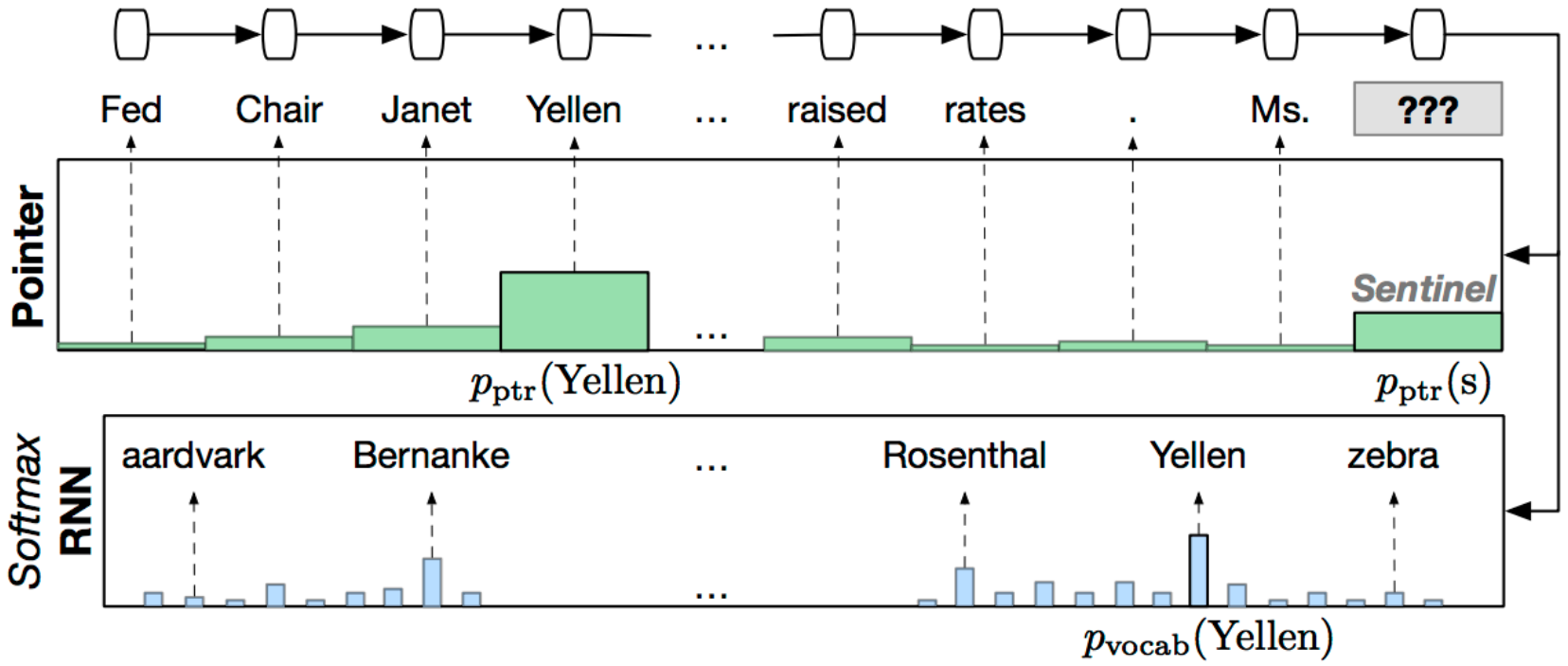
- Can only predict frequently seen classes
- Example: Language modeling where classes=words

- New words occur all the time during testing

- Solution: Combine softmax with pointers to context words!
- Work by Stephen Merity et al. 2016 (Released next week :)

@Smerity

Pointer sentinel mixture models



Language Model Evaluation

- Perplexity: 2^J where: $J = -\frac{1}{T}\sum_{t=1}^{T}\sum_{j=1}^{|V|} y_{t,j} \log \hat{y}_{t,j}$
- Lower is better
- Results with normal RNNs plus count-based models →
- Mikolov 2010

Model	PPL
GT3	165.2
KN5	147.8
KN5+cache	133.1
Structured LM (Chelba)	148.9
Structured LM (Roark)	137.2
Structured LM (Filimonov)	127.2
Random Forest (Peng Xu)	131.9
PAQ8o10t	131.1
Syntactic NN (Emami, baseline KN4 141)	107
8xRNN static	105.4
8xRNN dynamic	104.5
static+dynamic	97.4
+KN5	93.9
+KN5(cache)	90.4
+Random forest (Peng Xu)	87.9
+Structured LM (Filimonov)	87.7

Lots of progress in last years

From 87 perplexity with 8 RNNs ensemble plus count-based methods to 70.9 with single end-to-end trainable neural model

Model	Parameters	Validation	Test
Zaremba et al. (2014) - LSTM (medium)	20M	86.2	82.7
Zaremba et al. (2014) - LSTM (large)	66M	82.2	78.4
Gal (2015) - Variational LSTM (medium, untied)	20M	81.9 ± 0.2	79.7 ± 0.1
Gal (2015) - Variational LSTM (medium, untied, MC)	20M	–	78.6 ± 0.1
Gal (2015) - Variational LSTM (large, untied)	66M	77.9 ± 0.3	75.2 ± 0.2
Gal (2015) - Variational LSTM (large, untied, MC)	66M	–	73.4 ± 0.0
Zilly et al. (2016) - Variational RHN	32M	72.8	71.3
Zoneout + Variational LSTM (medium)	20M	84.4	80.6
Pointer Sentinel-LSTM (medium)	21M	72.4	**70.9**

Section 3:
Dynamic memory networks

Current Research

Can all NLP tasks be seen as question answering problems?

QA Examples

I: Mary walked to the bathroom.
I: Sandra went to the garden.
I: Daniel went back to the garden.
I: Sandra took the milk there.
Q: Where is the milk?
A: garden
I: Everybody is happy.
Q: What's the sentiment?
A: positive

I: Jane has a baby in Dresden.
Q: What are the named entities?
A: Jane - person, Dresden - location
I: Jane has a baby in Dresden.
Q: What are the POS tags?
A: NNP VBZ DT NN IN NNP .
I: I think this model is incredible
Q: In French?
A: Je pense que ce modèle est incroyable.

Goal

A joint model for general QA

First Major Obstacle

- For NLP no single model **architecture** with consistent state of the art results across tasks

Task	State of the art model
Question answering (babI)	Strongly Supervised MemNN (Weston et al 2015)
Sentiment Analysis (SST)	Tree-LSTMs (Tai et al. 2015)
Part of speech tagging (PTB-WSJ)	Bi-directional LSTM-CRF (Huang et al. 2015)

Second Major Obstacle

- Fully joint multitask learning* is hard:
  - Usually restricted to lower layers
  - Usually helps only if tasks are related
  - Often hurts performance if tasks are not related

* meaning: same decoder/classifier
  and not only transfer learning

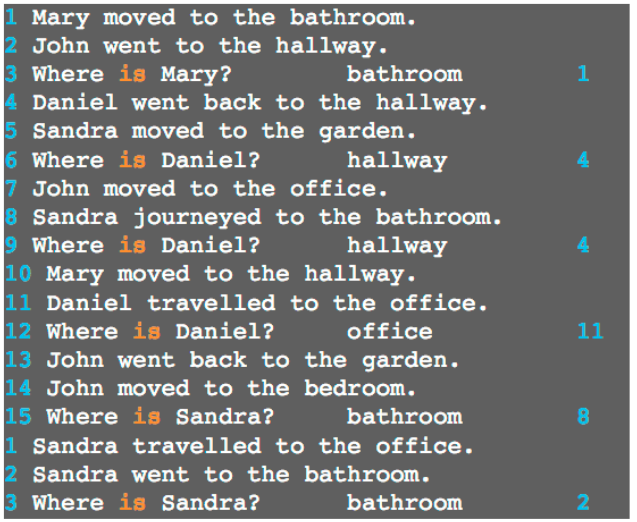
Tackling First Obstacle

Dynamic Memory Networks

An architecture for any QA task

High level idea for harder questions

- Imagine having to read an article, memorize it, then get asked various questions → Hard!
- You can't store everything in working memory
- **Optimal:** give you the input data, give you the question, allow as many glances as possible



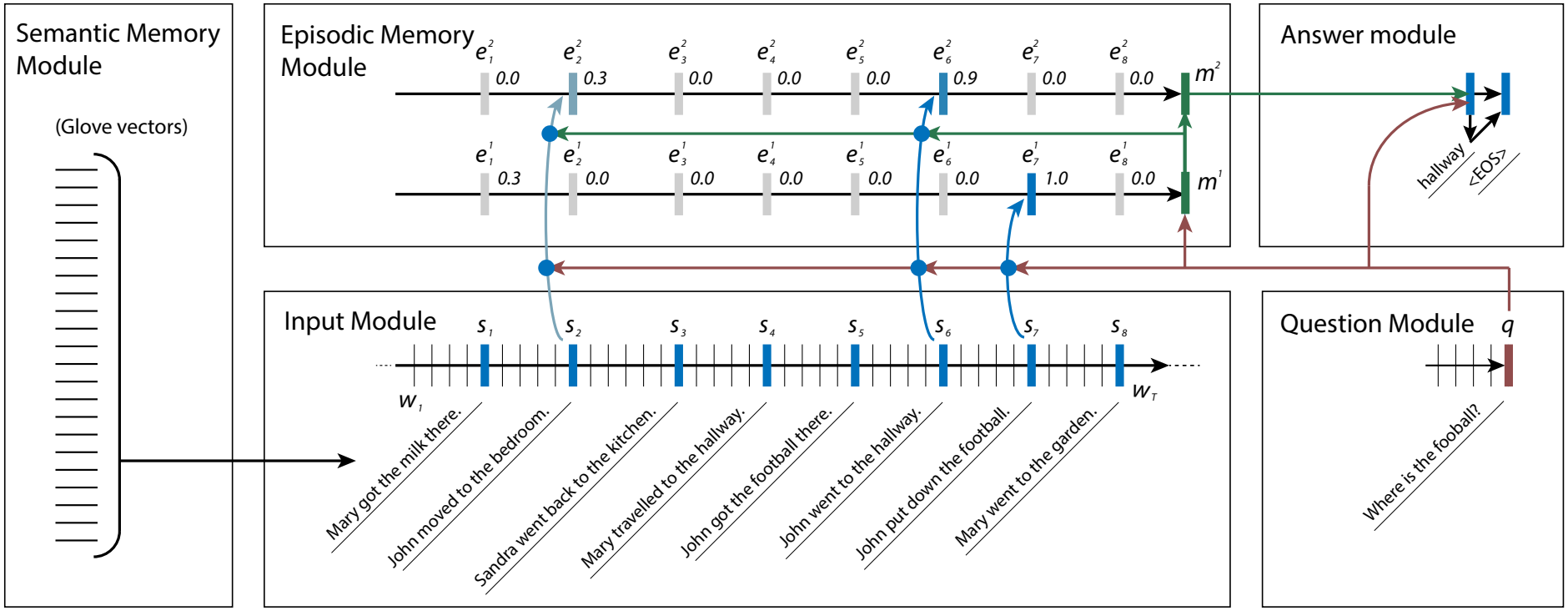
Basic lego block: GRU (defined before)

$$h_t = GRU(x_t, h_{t-1}) :$$

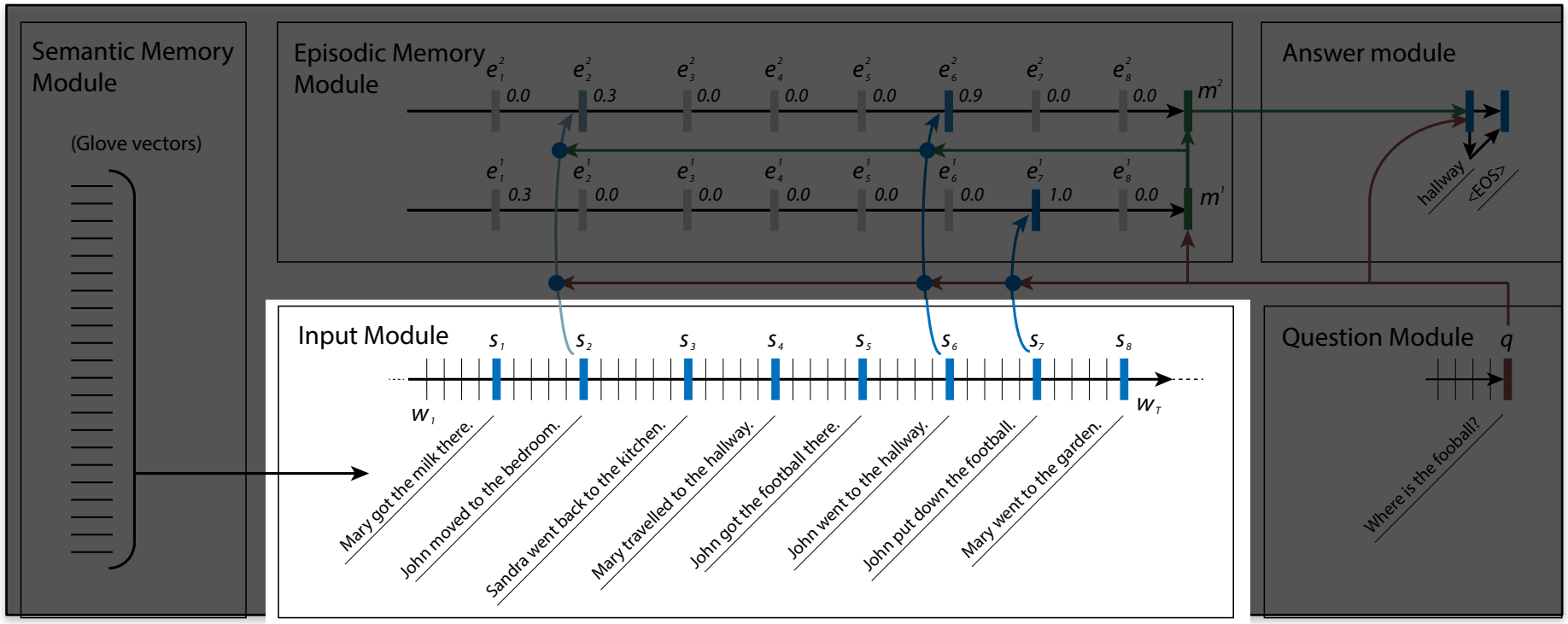
$$\begin{aligned}
z_t &= \sigma\left(W^{(z)} x_t + U^{(z)} h_{t-1} + b^{(z)}\right) \\
r_t &= \sigma\left(W^{(r)} x_t + U^{(r)} h_{t-1} + b^{(r)}\right) \\
\tilde{h}_t &= \tanh\left(W x_t + r_t \circ U h_{t-1} + b^{(h)}\right) \\
h_t &= z_t \circ h_{t-1} + (1 - z_t) \circ \tilde{h}_t,
\end{aligned}$$

Cho et al. 2014

Dynamic Memory Network

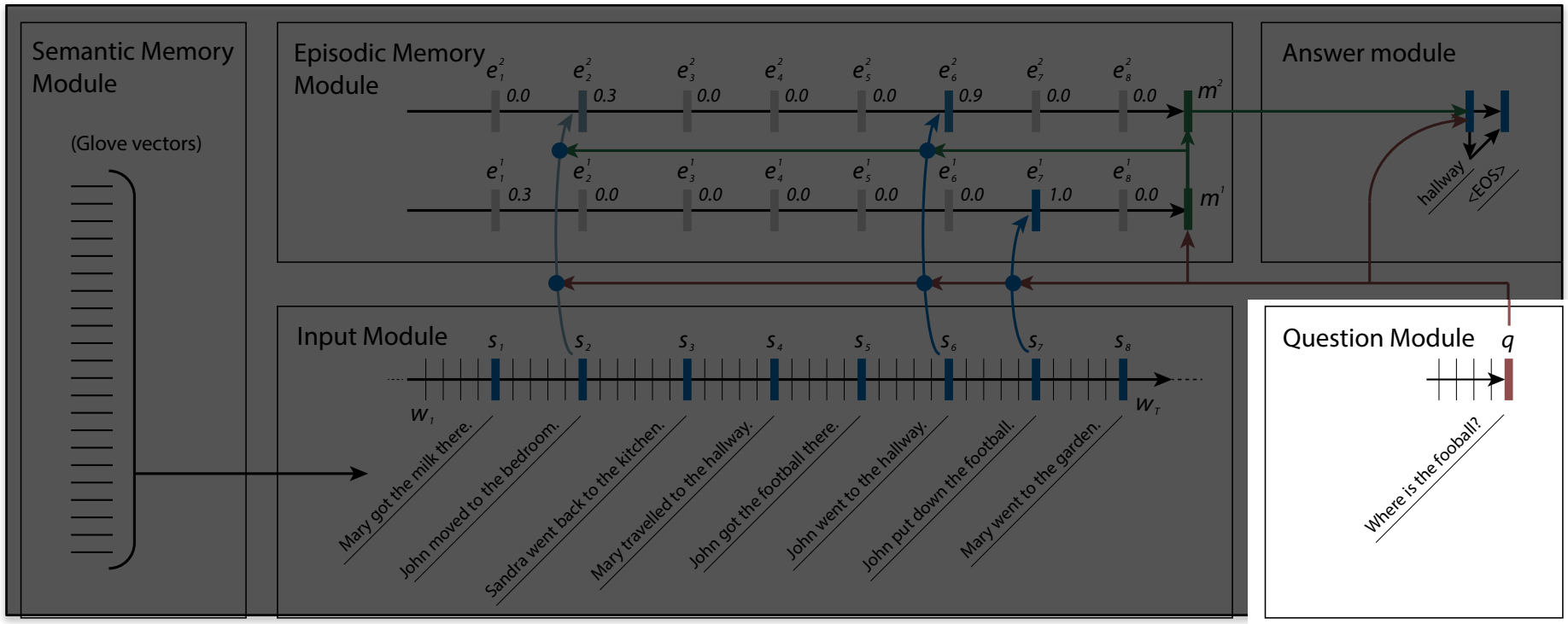


The Modules: Input



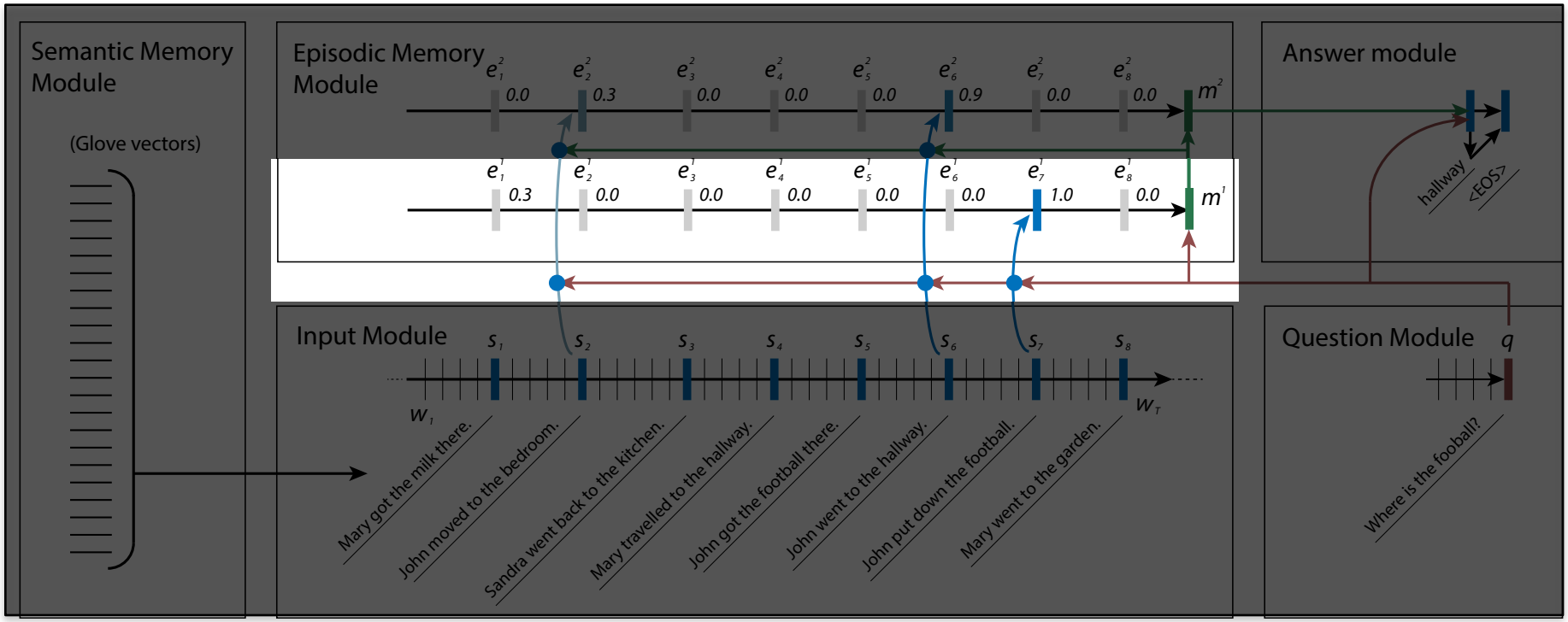
Standard GRU. The last hidden state of each sentence is accessible.

The Modules: Question



$$q_t = GRU(v_t, q_{t-1})$$

The Modules: Episodic Memory



$$h_i^t = g_i^t GRU\left(s_i, h_{i-1}^t\right) + \left(1 - g_i^t\right) h_{i-1}^t$$

Last hidden state: m^t

The Modules: Episodic Memory

- Gates are activated if sentence relevant to the question or memory

$$z_i^t = [s_i \circ q \; ; s_i \circ m^{t-1}; |s_i - q| \; ; |s_i - m^{t-1}|]$$

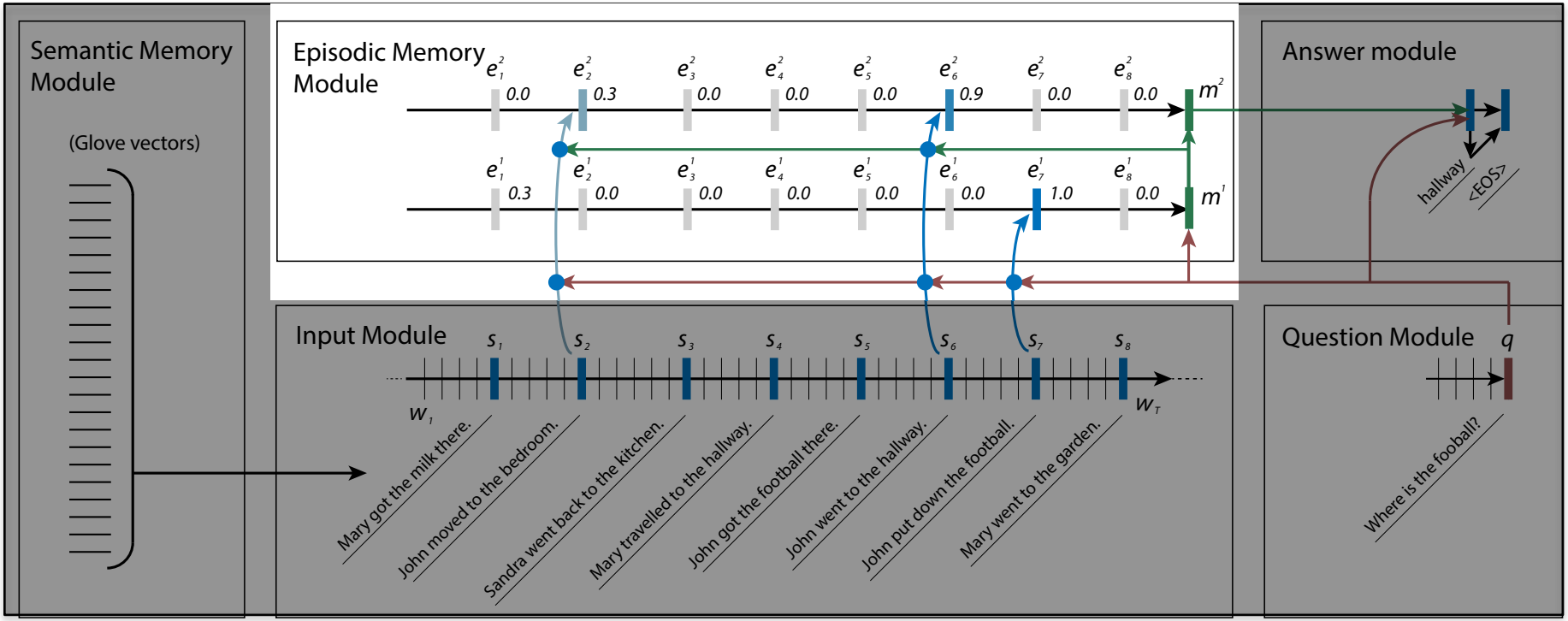
$$Z_i^t = W^{(2)} \tanh\left(W^{(1)} z_i^t + b^{(1)}\right) + b^{(2)}$$

$$g_i^t = \frac{\exp(Z_i^t)}{\sum_{k=1}^{M_i} \exp(Z_k^t)}$$

- When the end of the input is reached, the relevant facts are summarized in another GRU

The Modules: Episodic Memory

- If summary is insufficient to answer the question, repeat sequence over input



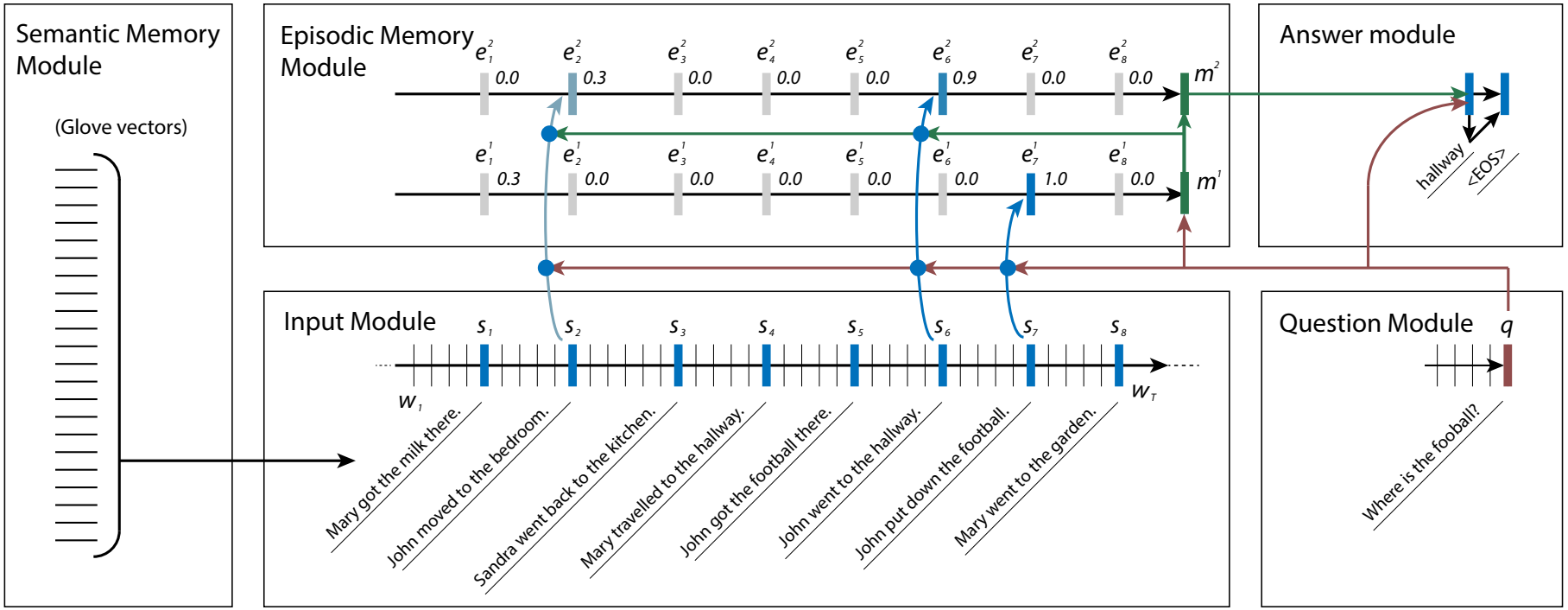
Inspiration from Neuroscience

- **Episodic memory** is the **memory** of autobiographical events (times, places, etc). A collection of past personal experiences that occurred at a particular time and place.

- The hippocampus, the seat of episodic memory in humans, is active during transitive inference
- In the DMN repeated passes over the input are needed for transitive inference

The Modules: Answer

$$a_t = \texttt{GRU}([y_{t-1}, q], a_{t-1}), \qquad y_t = softmax(W^{(a)} a_t)$$



Related work

- Sequence to Sequence (Sutskever et al. 2014)
- Neural Turing Machines (Graves et al. 2014)
- Teaching Machines to Read and Comprehend (Hermann et al. 2015)
- Learning to Transduce with Unbounded Memory (Grefenstette 2015)
- Structured Memory for Neural Turing Machines (Wei Zhang 2015)

- Memory Networks (Weston et al. 2015)
- End to end memory networks (Sukhbaatar et al. 2015)

→ More on these in Quoc's lecture

Comparison to MemNets

Similarities:

- MemNets and DMNs have input, scoring, attention and response mechanisms

Differences:

- For input representations MemNets use bag of word, nonlinear or linear embeddings that explicitly encode position
- MemNets iteratively run functions for attention and response

- **DMNs show that neural sequence models can be used for input representation, attention and response mechanisms**
  → naturally captures position and temporality
- Enables broader range of applications

Experiments: QA on babI (1k)

Task	MemNN	DMN	Task	MemNN	DMN
1: Single Supporting Fact	100	100	11: Basic Coreference	100	99.9
2: Two Supporting Facts	100	98.2	12: Conjunction	100	100
3: Three Supporting facts	100	95.2	13: Compound Coreference	100	99.8
4: Two Argument Relations	100	100	14: Time Reasoning	99	100
5: Three Argument Relations	98	99.3	15: Basic Deduction	100	100
6: Yes/No Questions	100	100	16: Basic Induction	100	99.4
7: Counting	85	96.9	17: Positional Reasoning	65	59.6
8: Lists/Sets	91	96.5	18: Size Reasoning	95	95.3
9: Simple Negation	100	100	19: Path Finding	36	34.5
10: Indefinite Knowledge	98	97.5	20: Agent's Motivations	100	100
			Mean Accuracy (%)	93.3	**93.6**

This still requires that relevant facts are marked during training to train the gates.

Experiments: Sentiment Analysis

- Stanford Sentiment Treebank

- Test accuracies:
- MV-RNN and RNTN: Socher et al. (2013)
- DCNN: Kalchbrenner et al. (2014)
- PVec: Le & Mikolov. (2014)
- CNN-MC: Kim (2014)
- DRNN: Irsoy & Cardie (2015)
- CT-LSTM: Tai et al. (2015)

Task	Binary	Fine-grained
MV-RNN	82.9	44.4
RNTN	85.4	45.7
DCNN	86.8	48.5
PVec	87.8	48.7
CNN-MC	88.1	47.4
DRNN	86.6	49.8
CT-LSTM	88.0	51.0
DMN	**88.6**	**52.1**

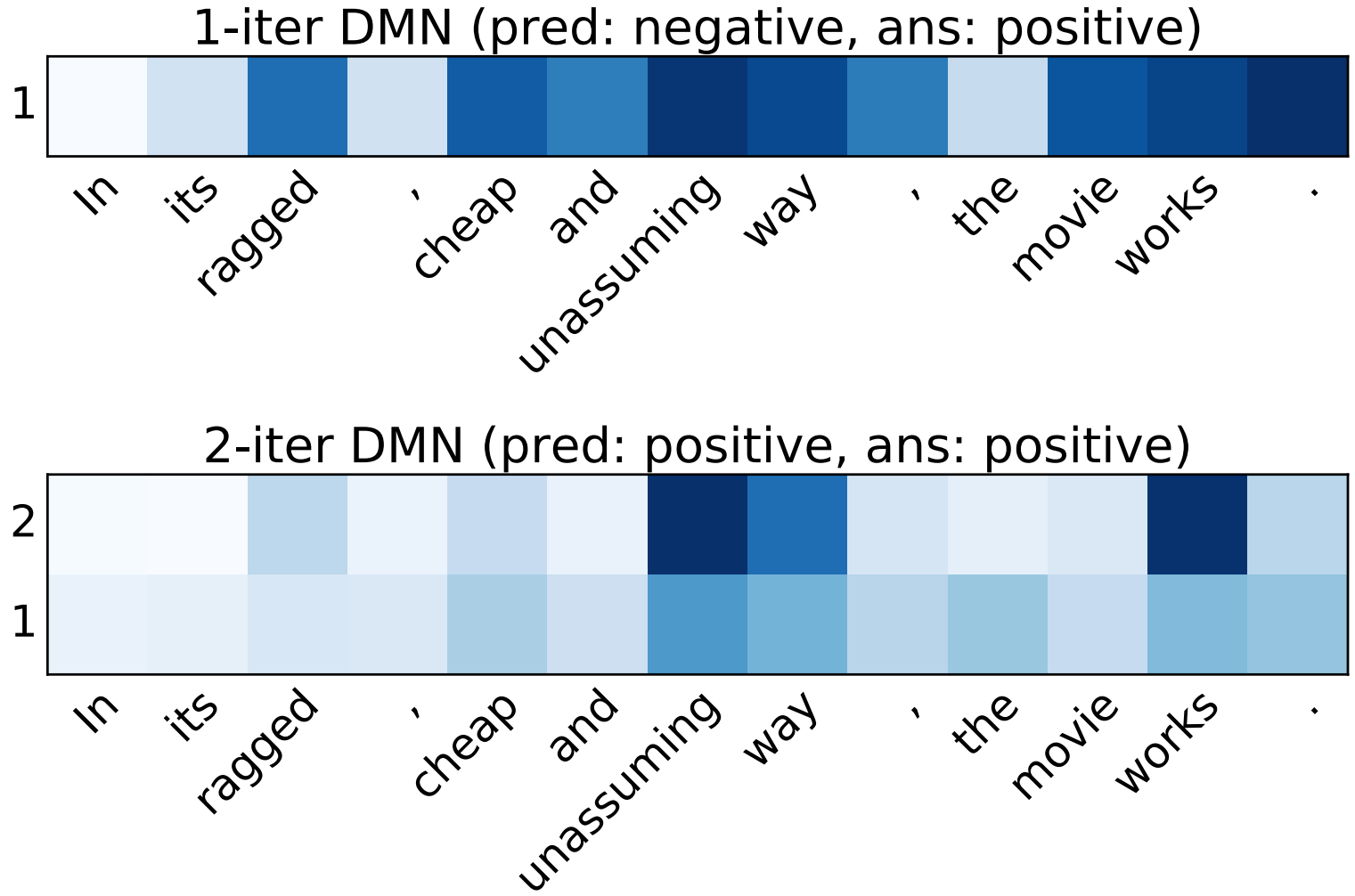
Analysis of Number of Episodes

- How many attention + memory passes are needed in the episodic memory?

Max passes	task 3 three-facts	task 7 count	task 8 lists/sets	sentiment (fine grain)
0 pass	0	48.8	33.6	50.0
1 pass	0	48.8	54.0	51.5
2 pass	16.7	49.1	55.6	**52.1**
3 pass	64.7	83.4	83.4	50.1
5 pass	**95.2**	**96.9**	**96.5**	N/A

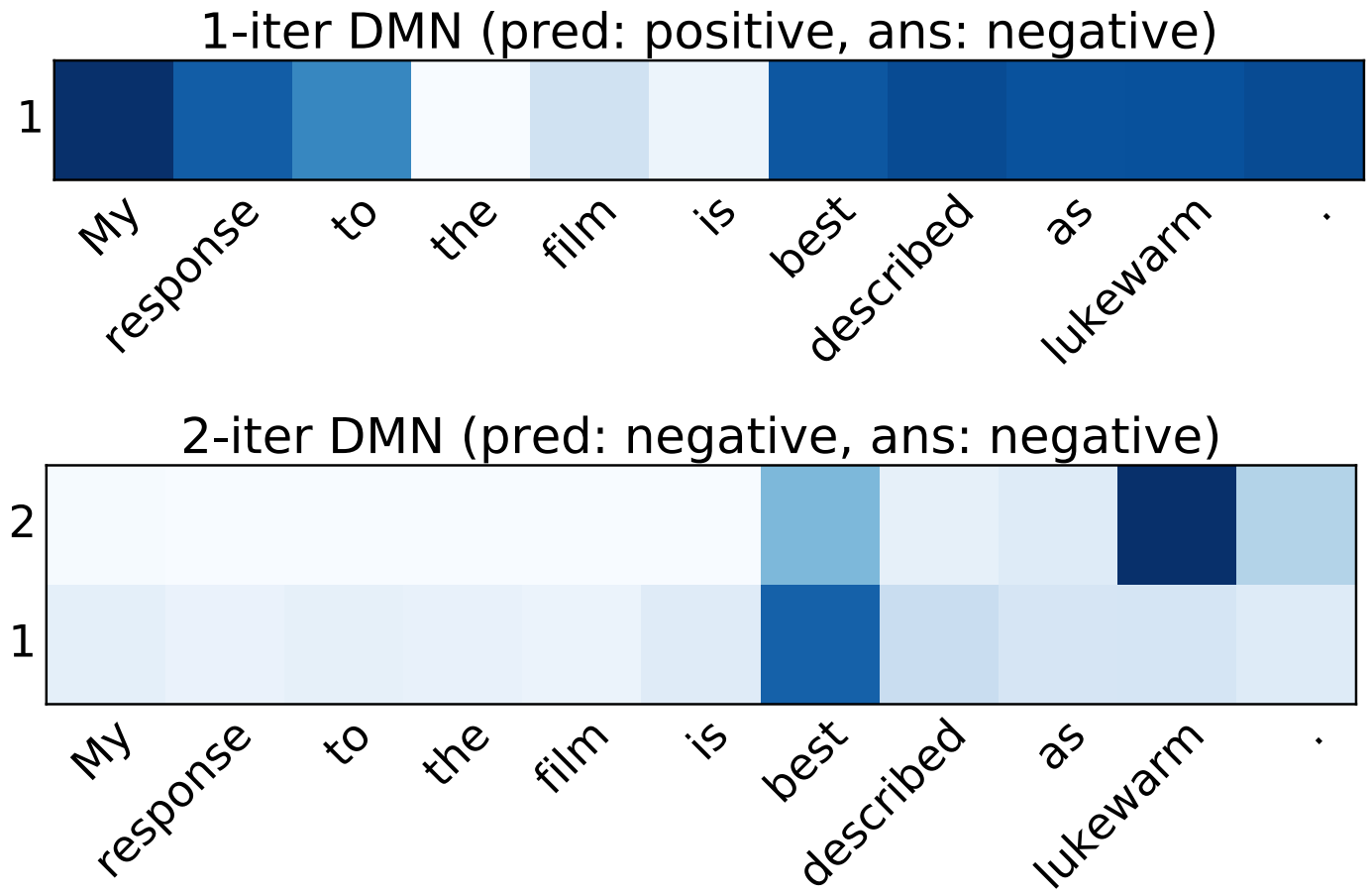
Analysis of Attention for Sentiment

- Sharper attention when 2 passes are allowed.
- Examples that are wrong with just one pass



Analysis of Attention for Sentiment

- Examples where full sentence context from first pass changes attention to words more relevant for final prediction

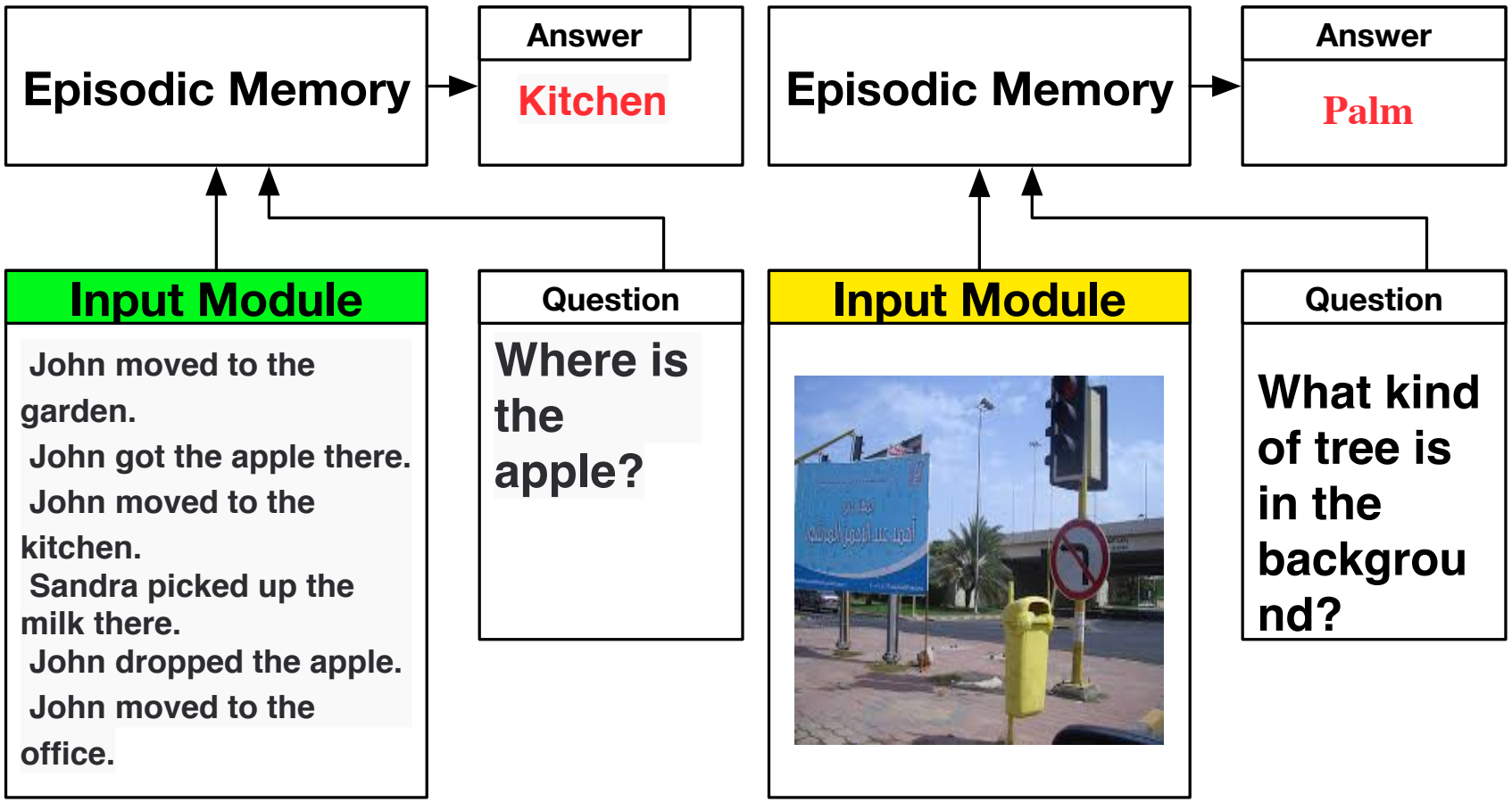


Experiments: POS Tagging

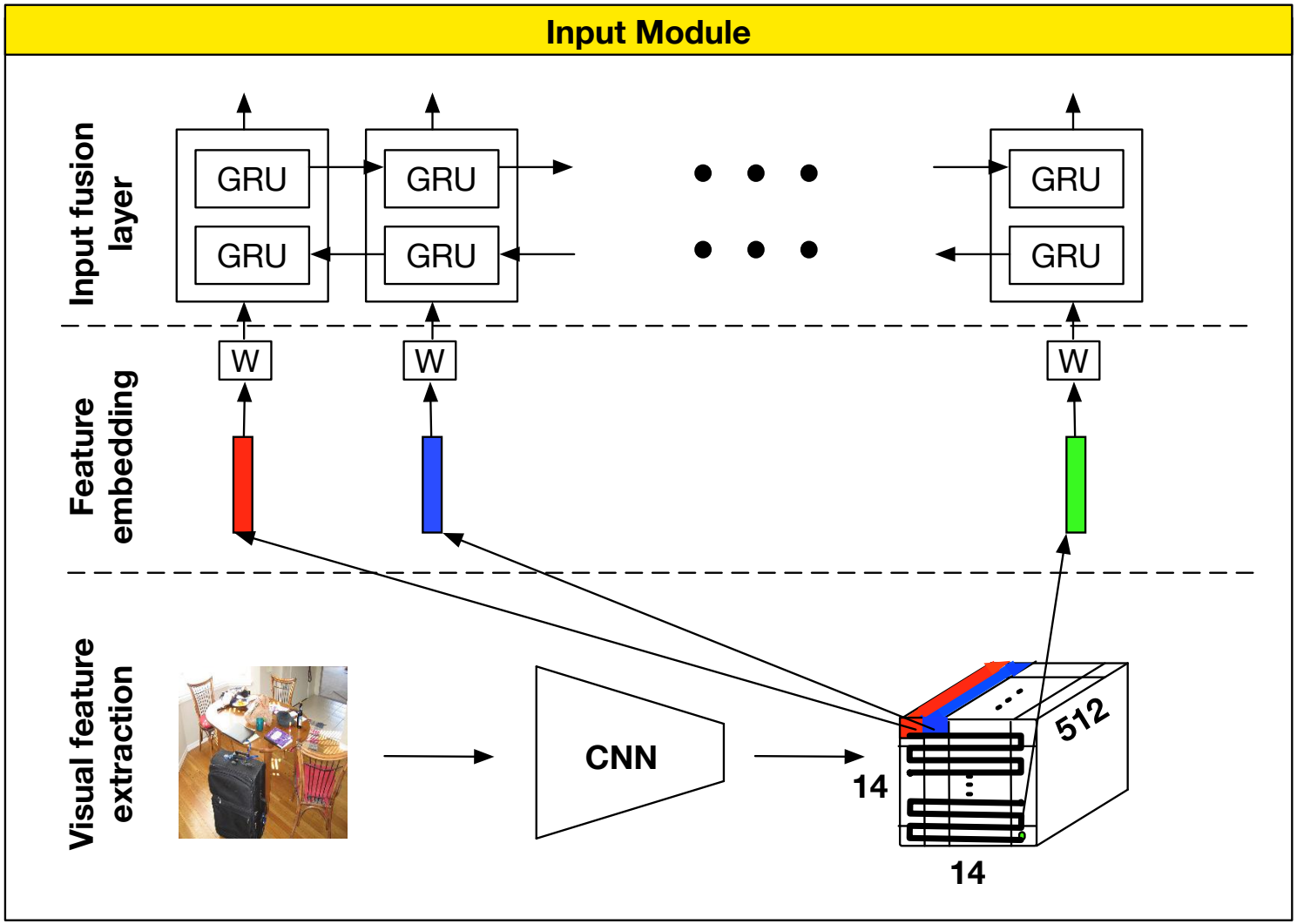
- PTB WSJ, standard splits
- Episodic memory does not require multiple passes, single pass enough

Model	SVMTool	Sogaard	Suzuki et al.	Spoustova et al.	SCNN	DMN
Acc (%)	97.15	97.27	97.40	97.44	97.50	**97.56**

Modularization Allows for Different Inputs



Input Module for Images



Accuracy: Visual Question Answering

VQA test-dev and test-standard:

- Antol et al. (2015)
- ACK Wu et al. (2015);
- iBOWIMG - Zhou et al. (2015);
- DPPnet - Noh et al. (2015); D-NMN - Andreas et al. (2016);
- SAN - Yang et al. (2015)

	test-dev				test-std
Method	All	Y/N	Other	Num	All
VQA					
Image	28.1	64.0	3.8	0.4	-
Question	48.1	75.7	27.1	36.7	-
Q+I	52.6	75.6	37.4	33.7	-
LSTM Q+I	53.7	78.9	36.4	35.2	54.1
ACK	55.7	79.2	40.1	36.1	56.0
iBOWIMG	55.7	76.5	42.6	35.0	55.9
DPPnet	57.2	80.7	41.7	37.2	57.4
D-NMN	57.9	80.5	43.1	37.4	58.0
SAN	58.7	79.3	46.1	36.6	58.9
DMN+	**60.3**	80.5	48.3	36.8	**60.4**

Attention Visualization


What is the main color on the bus ?

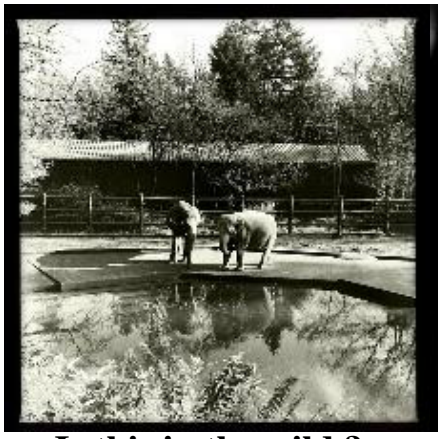
Answer: blue


What type of trees are in the background ?

Answer: pine


How many pink flags are there ?

Answer: 2




Is this in the wild ?

Answer: no

Attention Visualization

Which man is dressed more flamboyantly ? **Answer: right**

Who is on both photos ? **Answer: girl**

What time of day was this picture taken ? **Answer: night**

What is the boy holding ? **Answer: surfboard**

Attention Visualization

What is this sculpture made out of ?

Answer: metal

What color are the bananas ?

Answer: green

What is the pattern on the cat ' s fur on its tail ?

Answer: stripes

Did the player hit the ball ?

Answer: yes

Live Demo


Dynamic Memory Network by MetaMind
Screen Shot 2015-12-12 at 4.10.04 AM.png
3.9 MB
Done
Undo Image Upload
Question
What color is the building that has a clock ?
Run DMN
VQA sample

What is the girl holding ?

tennis racket

What is the girl doing ?

playing tennis

Is the girl wearing a hat ?

yes

What is the girl wearing ?

shorts

What is the color of the ground ?

brown

What color is the ball ?

yellow

What color is her skirt ?

white

What did the girl just hit ?

tennis ball

Summary

- Word vectors and RNNs are building blocks
- Most NLP tasks can be reduced to QA
- DMN accurately solves variety of QA tasks