Capturing Dependency Syntax with "Deep" Sequential Models

Yoav Goldberg

DepLing 2017

BIU NLP

Bar-Ilan University
אוניברסיטת בר-אילן

Capturing Dependency Syntax with "Deep" Sequential Models

Yoav Goldberg

DepLing 2017

Capturing Dependency Syntax with "Deep" Sequential Models

Yoav Goldberg

DepLing 2017

Eva's talk: "deep" sentential structure

B I U
N L P

Bar-Ilan University
אוניברסיטת בר-אילן

Deep Learning

Deep Learning

IT LEARNS ON ITS OWN.

IT WORKS LIKE THE BRAIN.

IT CAN DO ANYTHING.

My experience with Deep Learning for Language

``I'm sorry Dave, I'm afraid I can't do that."

(not in the scary sense)

My experience with Deep Learning for Language

- With proper tools, easy to produce "innovative" models.
- Not so easy to get good results.
- With Feed-forward nets, hard to beat linear models w/ human engineered feature combinations.
- On 20-newsgroups, NaiveBayes+TfIdf wins over deep Feed-forward-nets and ConvNets.

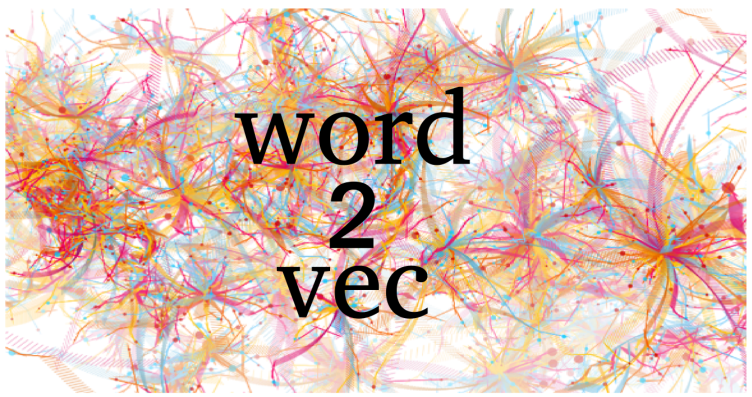
My experience with Deep Learning for Language

- With proper tools, easy to produce "innovative" models.
- Not so easy to get good results.
- With Feed-forward nets, hard to beat linear models w/ human engineered feature combinations.
- On 20-newsgroups, NaiveBayes+TfIdf wins over deep Feed-forward-nets and ConvNets.
- Semi-sup learning sort-of easy with word-embeddings.

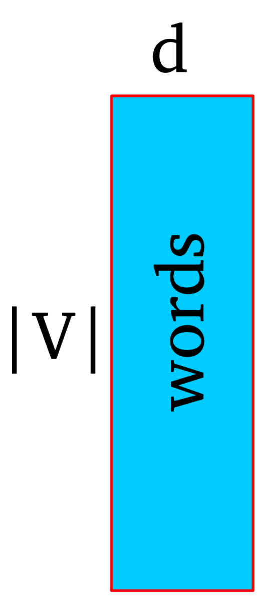
word2vec

Text

feed in text



wait a few hours



dog = (0.12,-0.32,0.92,0.43,-0.3 ...)
cat = (0.15,-0.29,0.90,0.39,-0.32 ...)

chair = (0.8,0.9,-0.76,0.29,0.52 ...)

get a |V|xd matrix W where each row is a vector for a word

- dog
  - cat, dogs, dachshund, rabbit, puppy, poodle, rottweiler, mixed-breed, doberman, pig
- sheep
  - cattle, goats, cows, chickens, sheeps, hogs, donkeys, herds, shorthorn, livestock
- november
  - october, december, april, june, february, july, september, january, august, march
- jerusalem
  - tiberias, jaffa, haifa, israel, palestine, nablus, damascus katamon, ramla, safed
- teva
  - pfizer, schering-plough, novartis, astrazeneca, glaxosmithkline, sanofi-aventis, mylan, sanofi, genzyme, pharmacia

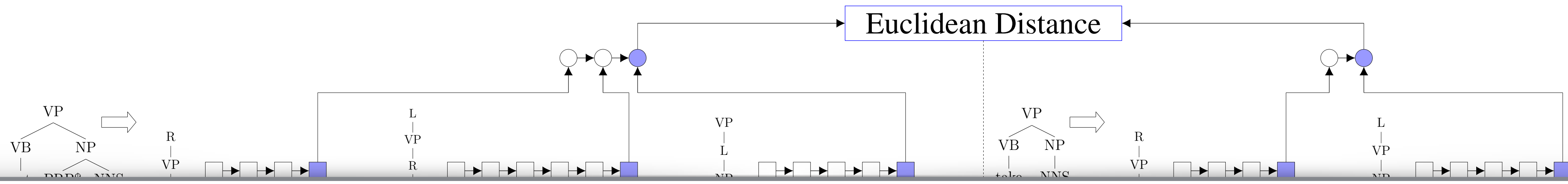
My experience with Deep Learning for Language

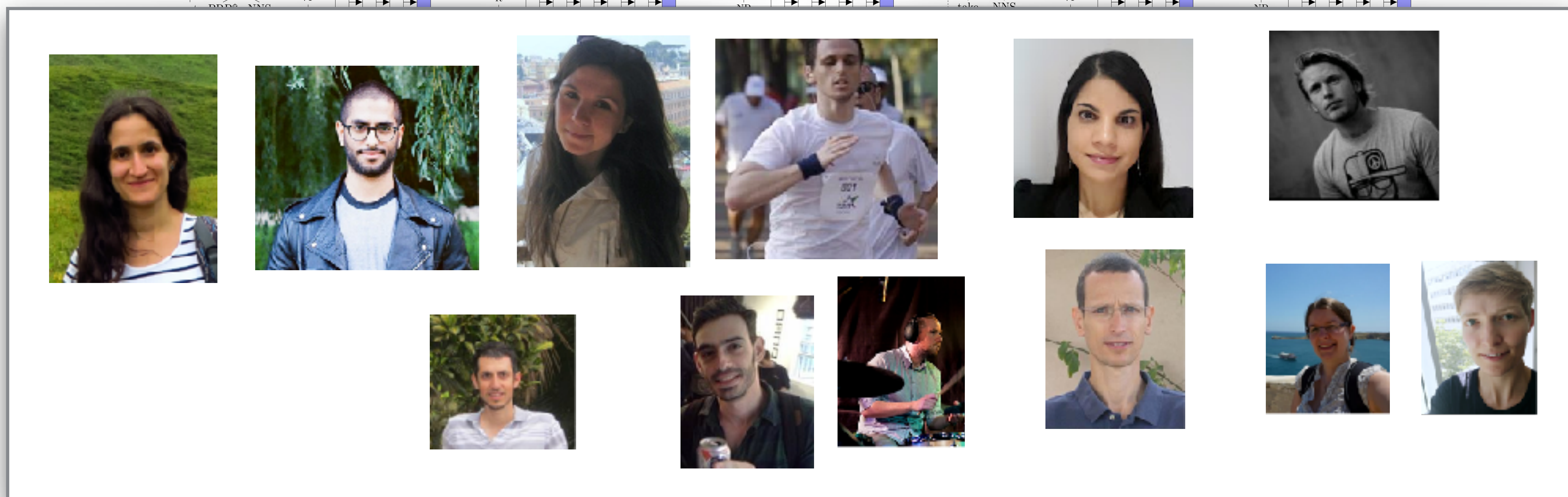
- With proper tools, easy to produce "innovative" models.
- Not so easy to get good results.
- With Feed-forward nets, hard to beat linear models w/ human engineered feature combinations.
- On 20-newsgroups, NaiveBayes+TfIdf wins over deep Feed-forward-nets and ConvNets.
- Semi-sup learning sort-of easy with word-embeddings.

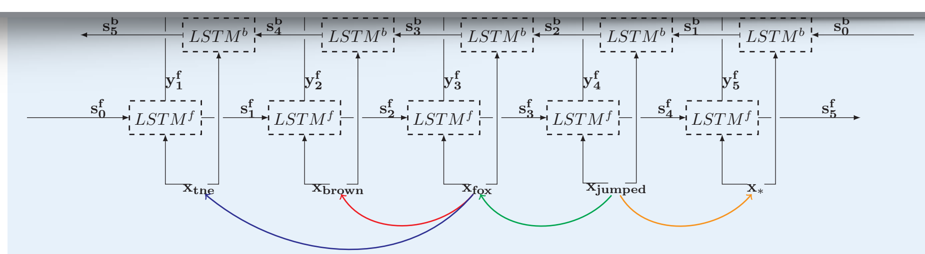
My experience with Deep Learning for Language

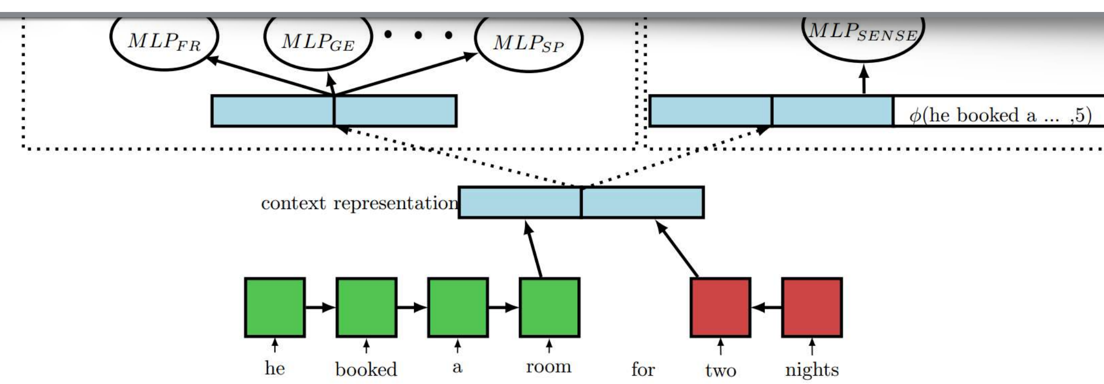
- With proper tools, easy to produce "innovative" models.
- Not so easy to get good results.
- With Feed-forward nets, hard to beat linear models w/ human engineered feature combinations.
- On 20-newsgroups, NaiveBayes+TfIdf wins over deep Feed-forward-nets and ConvNets.
- Semi-sup learning sort-of easy with word-embeddings.
- **RNNs (in particular LSTMs) are really really cool.**

Doing stuff with LSTMs

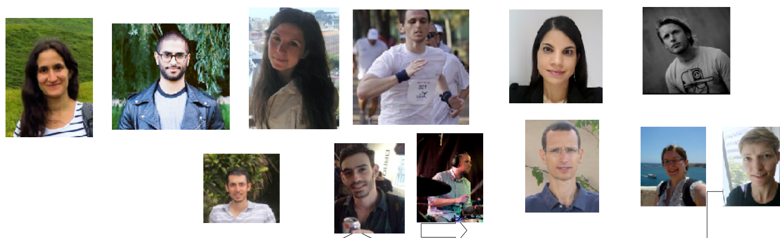


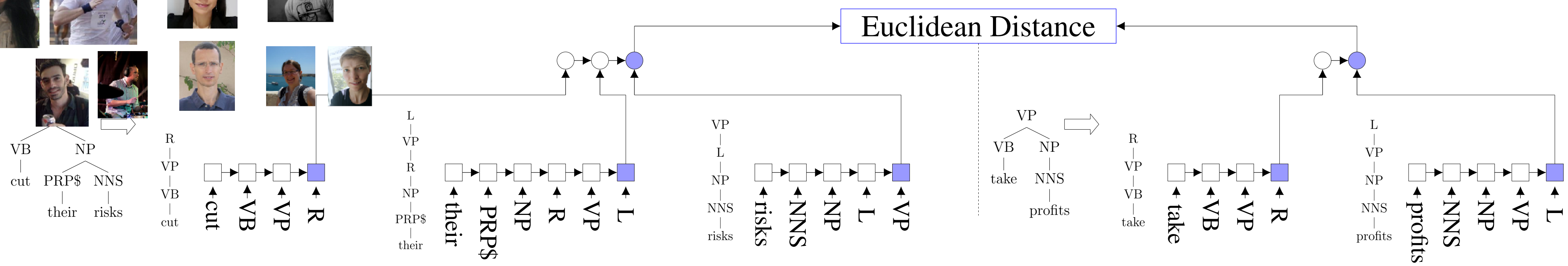


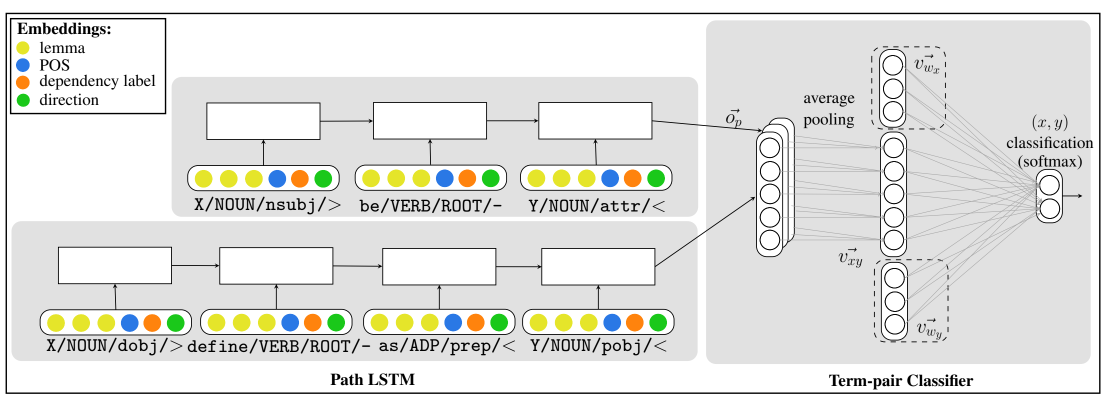


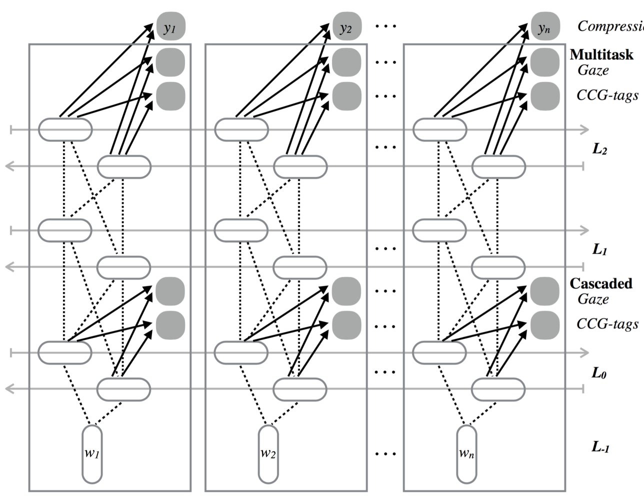


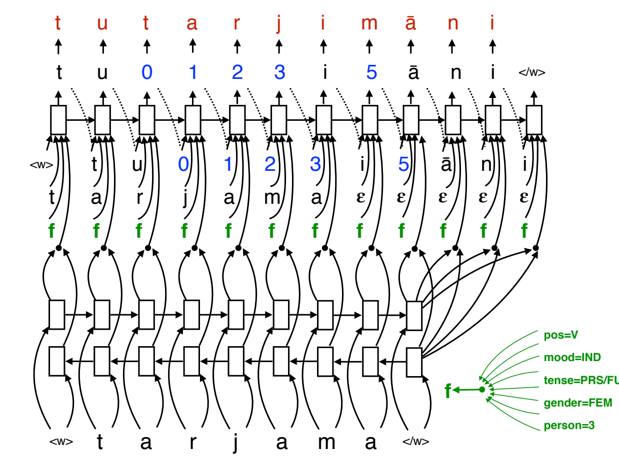
Doing stuff with LSTMs

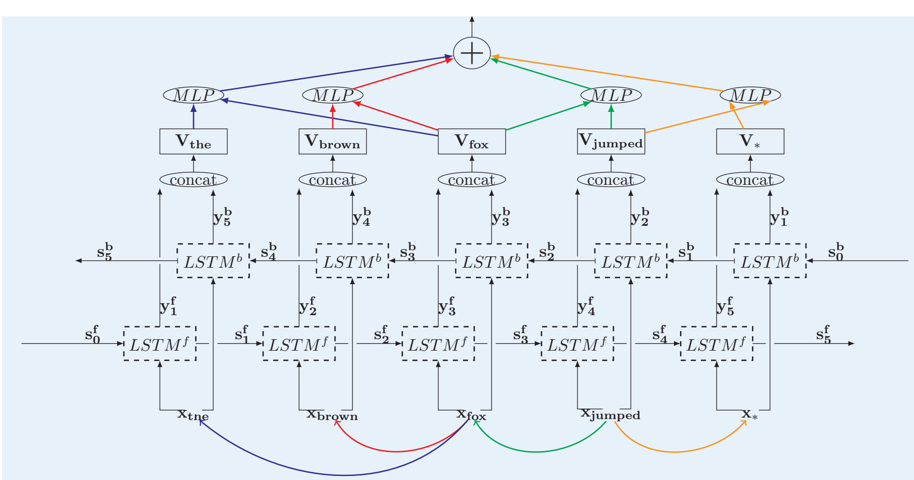


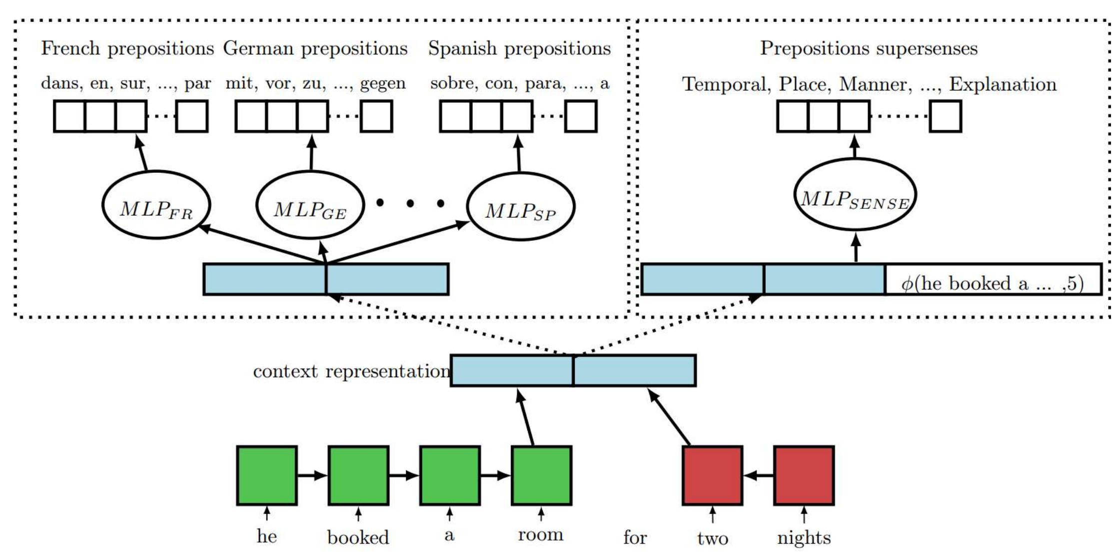








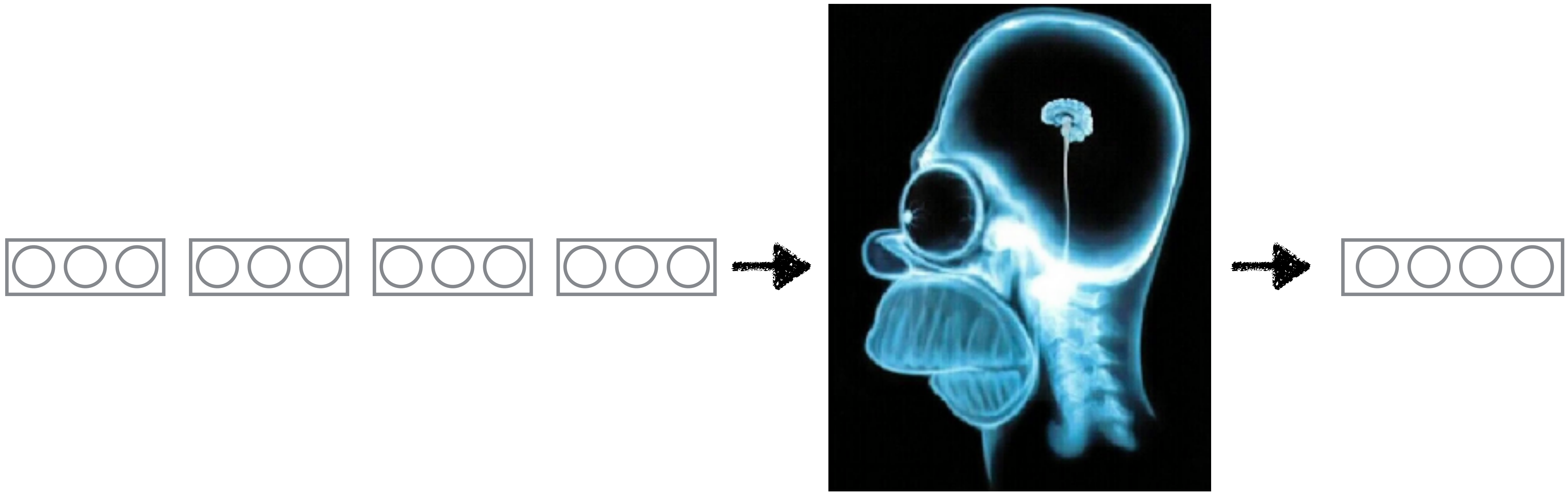




RNNS/LSTMs and Syntax

Brief intro to RNNs

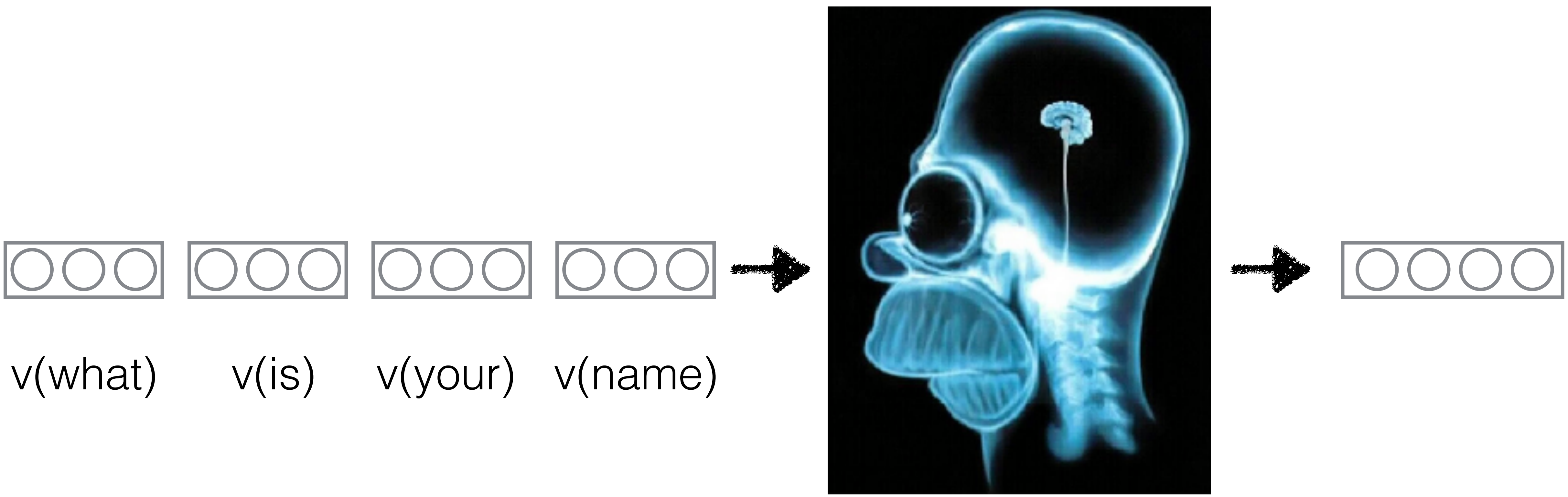
Recurrent Neural Networks



- Very strong models of sequential data.

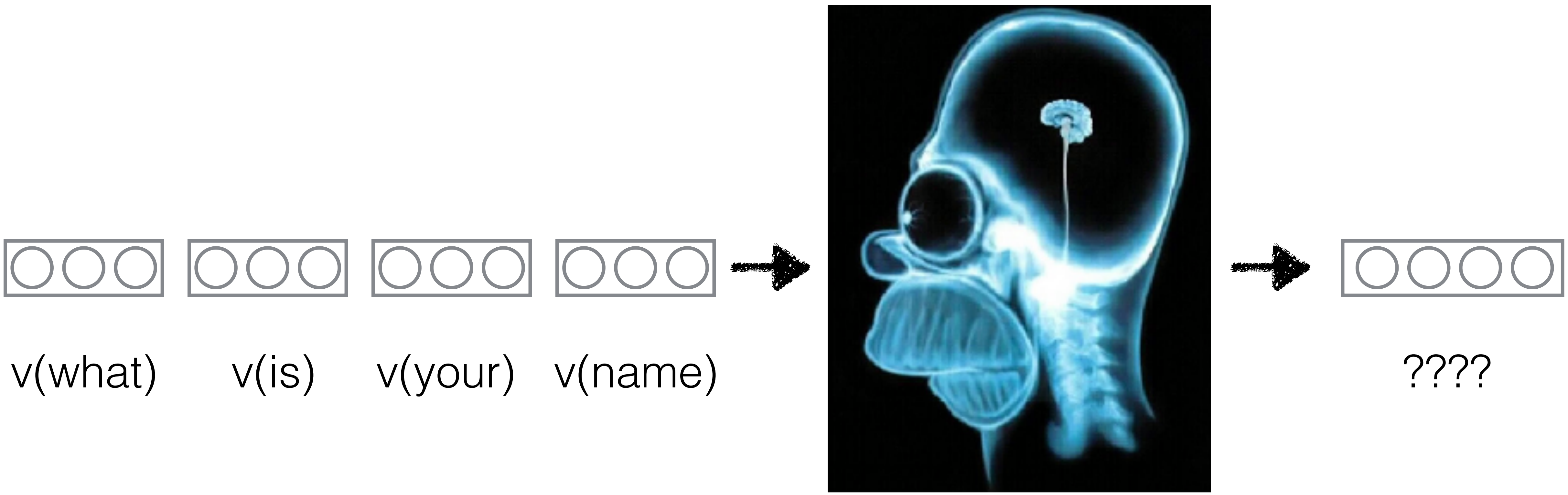
- Function from *n* vectors to a single vector.

Recurrent Neural Networks



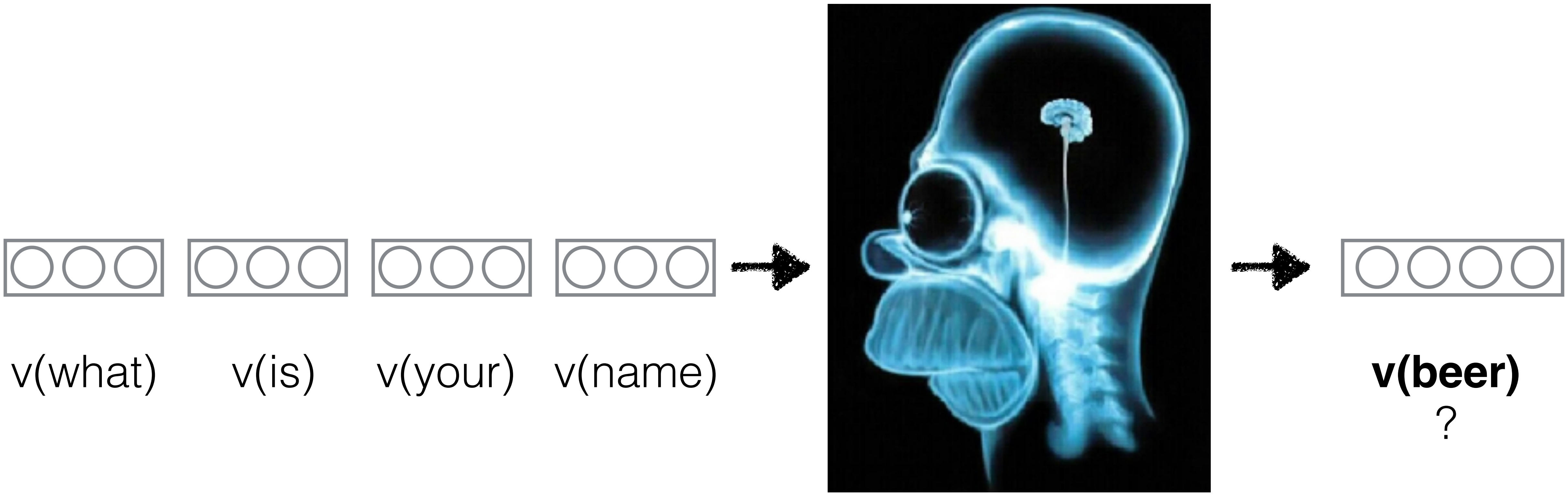
- Very strong models of sequential data.
- Function from *n* vectors to a single vector.

Recurrent Neural Networks



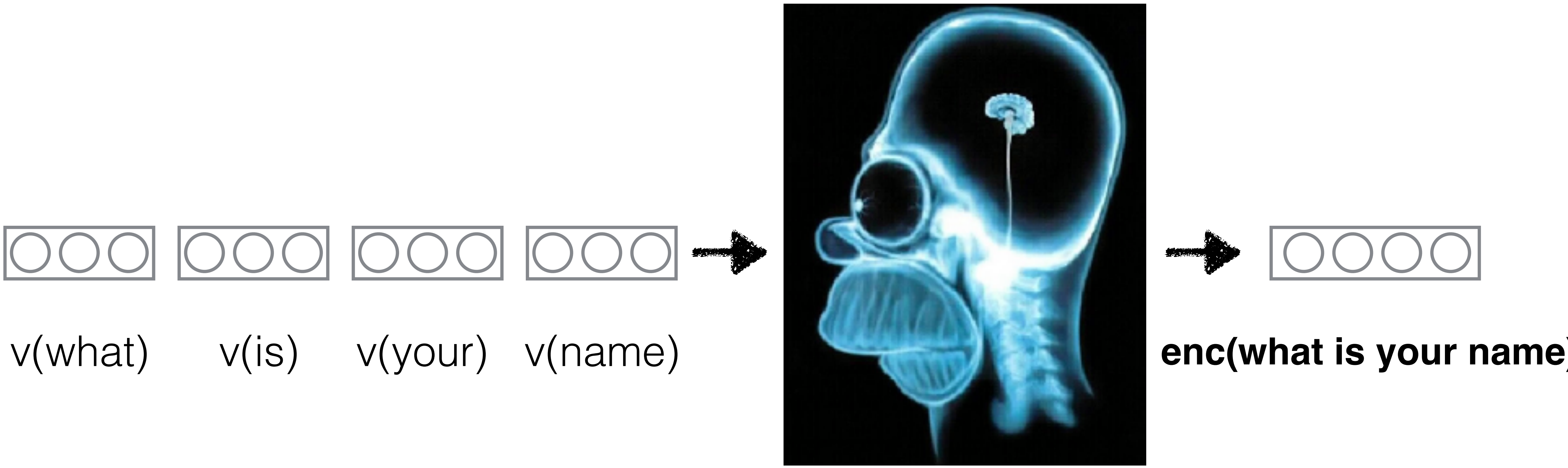
- Very strong models of sequential data.
- Function from *n* vectors to a single vector.

Recurrent Neural Networks



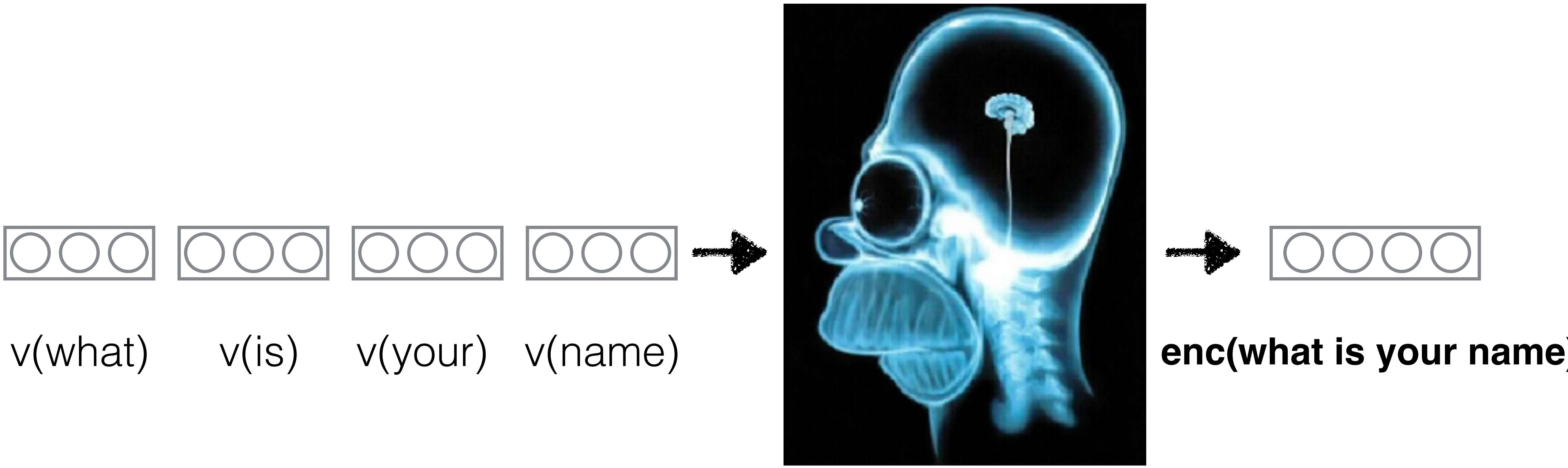
- Very strong models of sequential data.
- Function from *n* vectors to a single vector.

Recurrent Neural Networks



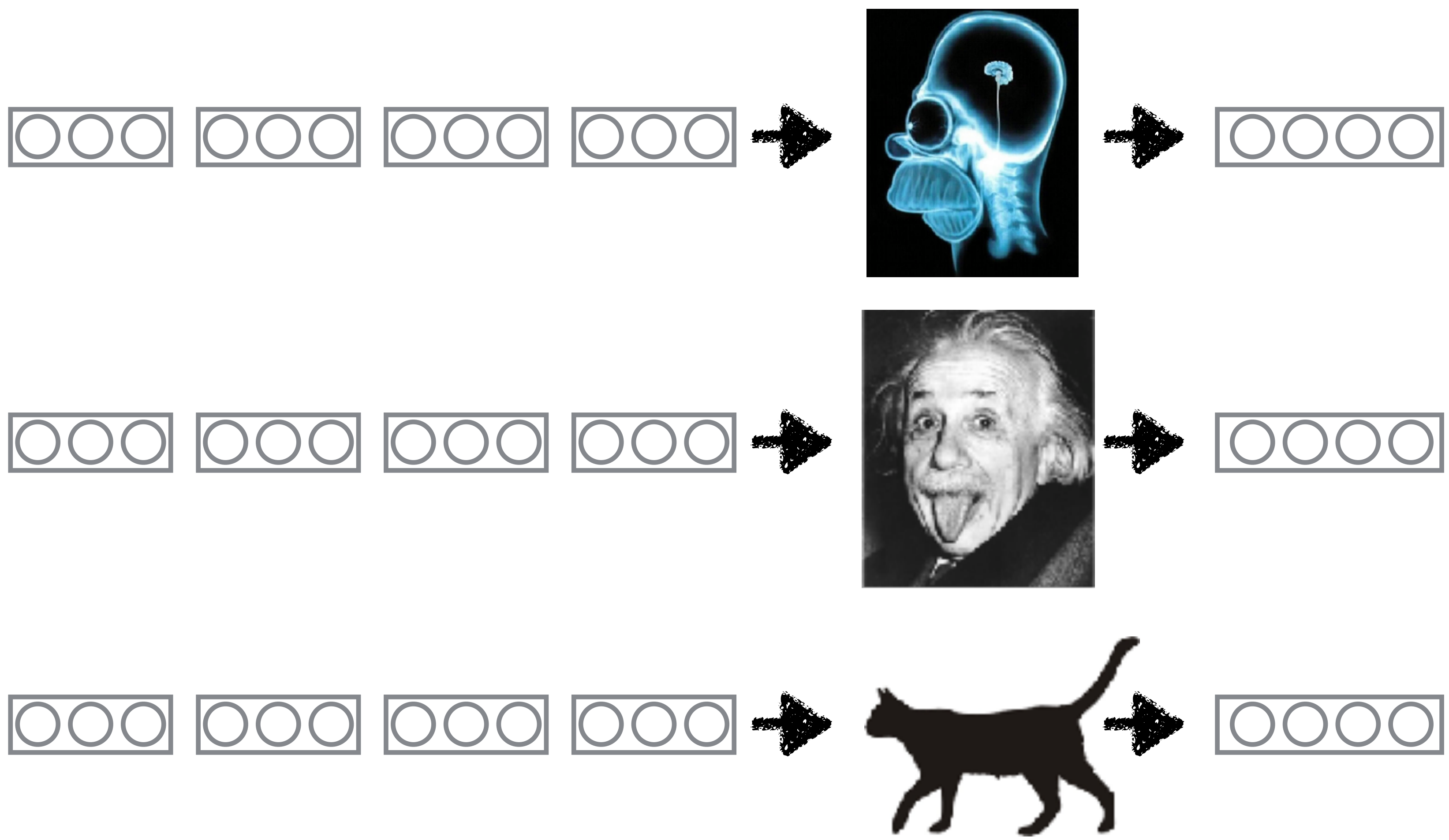
- Very strong models of sequential data.
- Function from *n* vectors to a single vector.

Recurrent Neural Networks



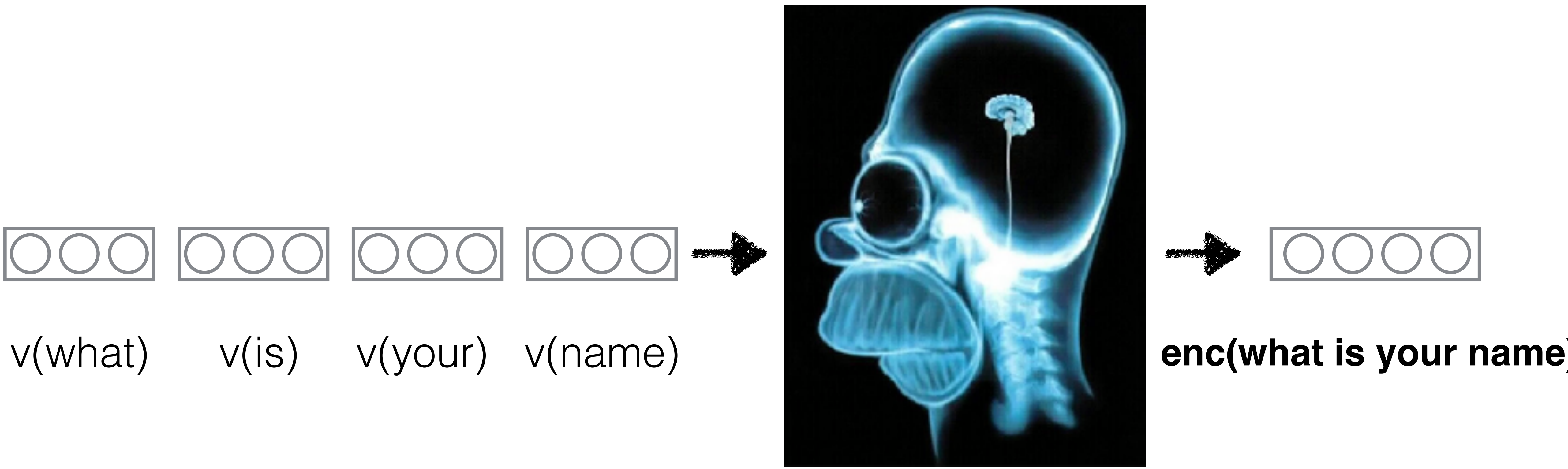
- Very strong models of sequential data.
- **Trainable** function from *n* vectors to a single vector.

Recurrent Neural Networks



- There are different variants (implementations).
- We'll focus on the interface level.

Recurrent Neural Networks



- Very strong models of sequential data.
- **Trainable** function from *n* vectors to a single vector.

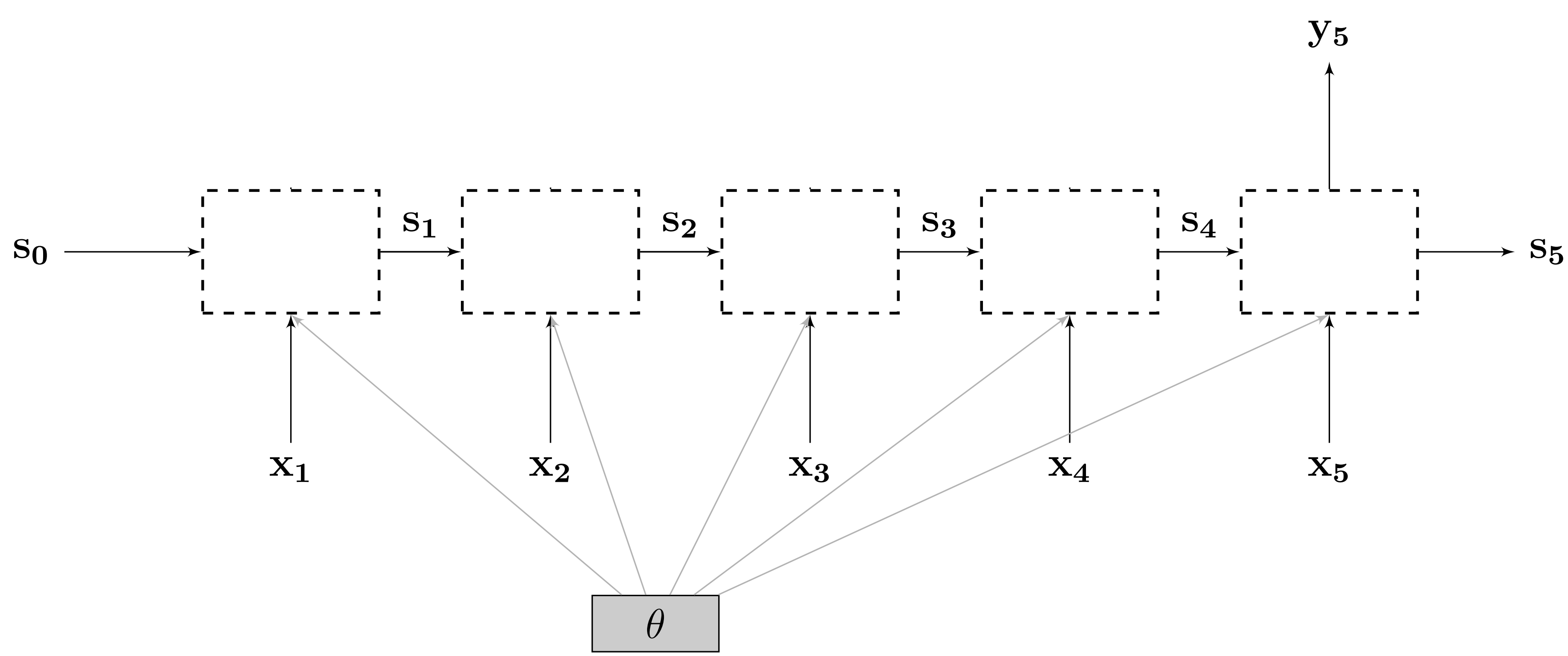
Recurrent Neural Networks

$$RNN(\mathbf{s_0}, \mathbf{x_{1:n}}) = \mathbf{s_n}, \mathbf{y_n}$$

$$\mathbf{x_i} \in \mathbb{R}^{d_{in}},\ \mathbf{y_i} \in \mathbb{R}^{d_{out}},\ \mathbf{s_i} \in \mathbb{R}^{f(d_{out})}$$

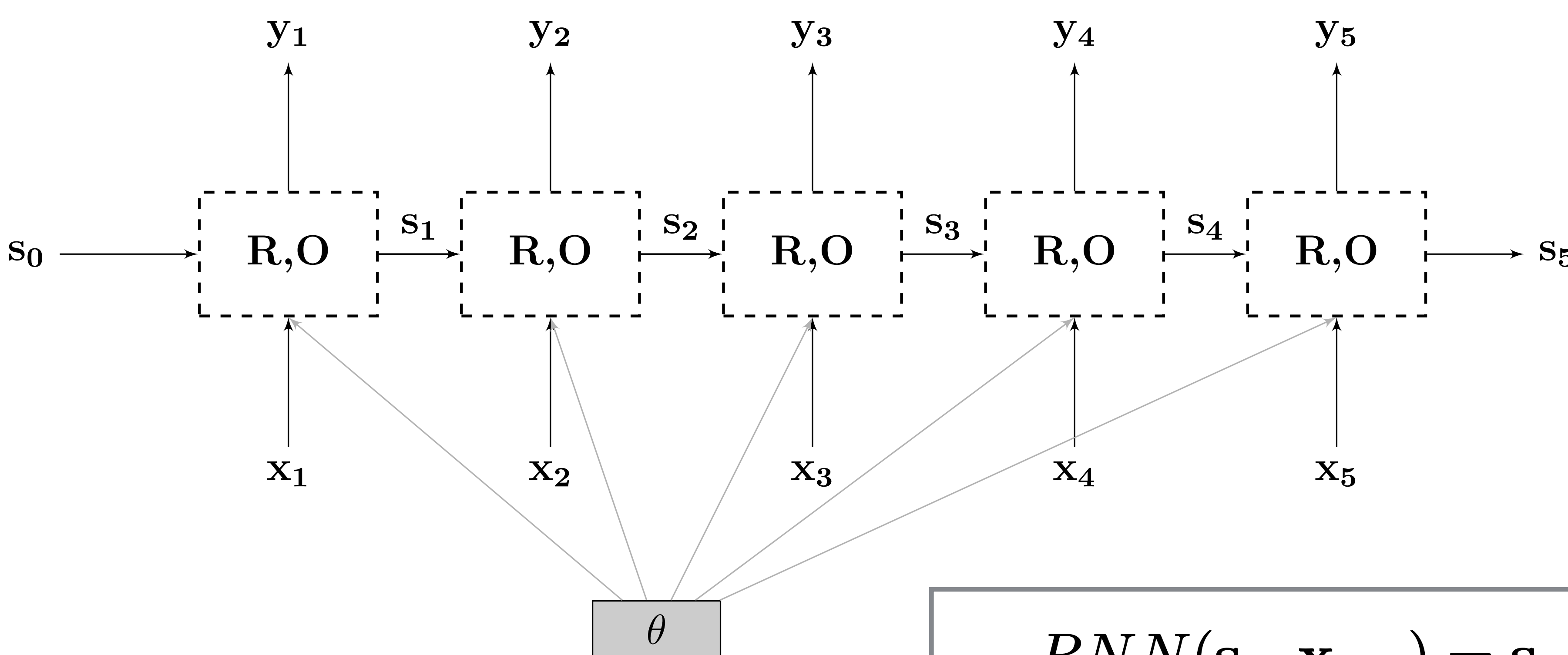
- Very strong models of sequential data.
- **Trainable** function from *n* vectors to a single* vector.

Recurrent Neural Networks



- Input vectors $\mathbf{x_{1:i}}$, output vector $\mathbf{y_i}$
- The output vector $\mathbf{y_i}$ depends on **all** inputs $\mathbf{x_{1:i}}$

Recurrent Neural Networks



$$RNN(\mathbf{s_0}, \mathbf{x_{1:n}}) = \mathbf{s_n}, \mathbf{y_n}$$

$$\mathbf{s_i} = R(\mathbf{s_{i-1}}, \mathbf{x_i})$$

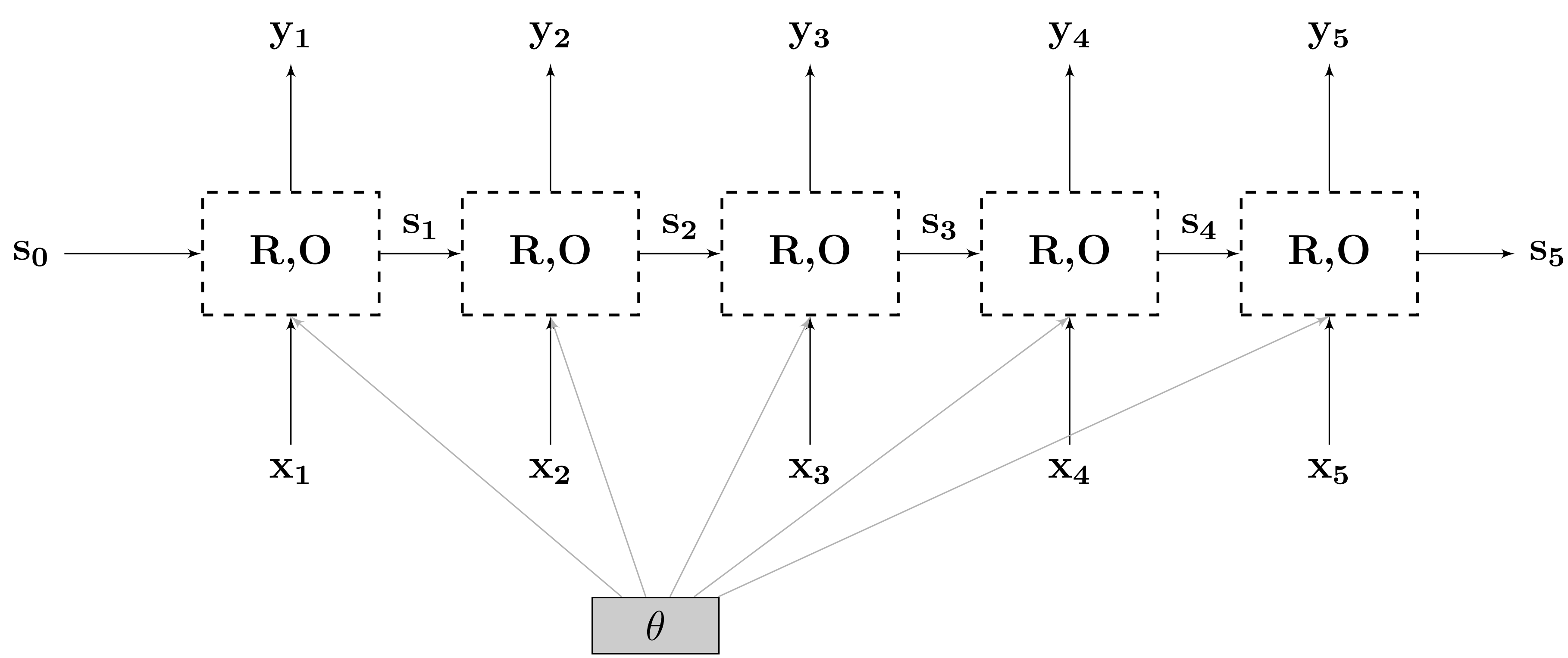
$$\mathbf{y_i} = O(\mathbf{s_i})$$

$$\mathbf{x_i} \in \mathbb{R}^{d_{in}},\ \mathbf{y_i} \in \mathbb{R}^{d_{out}},\ \mathbf{s_i} \in \mathbb{R}^{f(d_{out})}$$

- Recursively defined.
- There's a vector $\mathbf{y_i}$ for every prefix $\mathbf{x_{1:i}}$

Recurrent Neural Networks

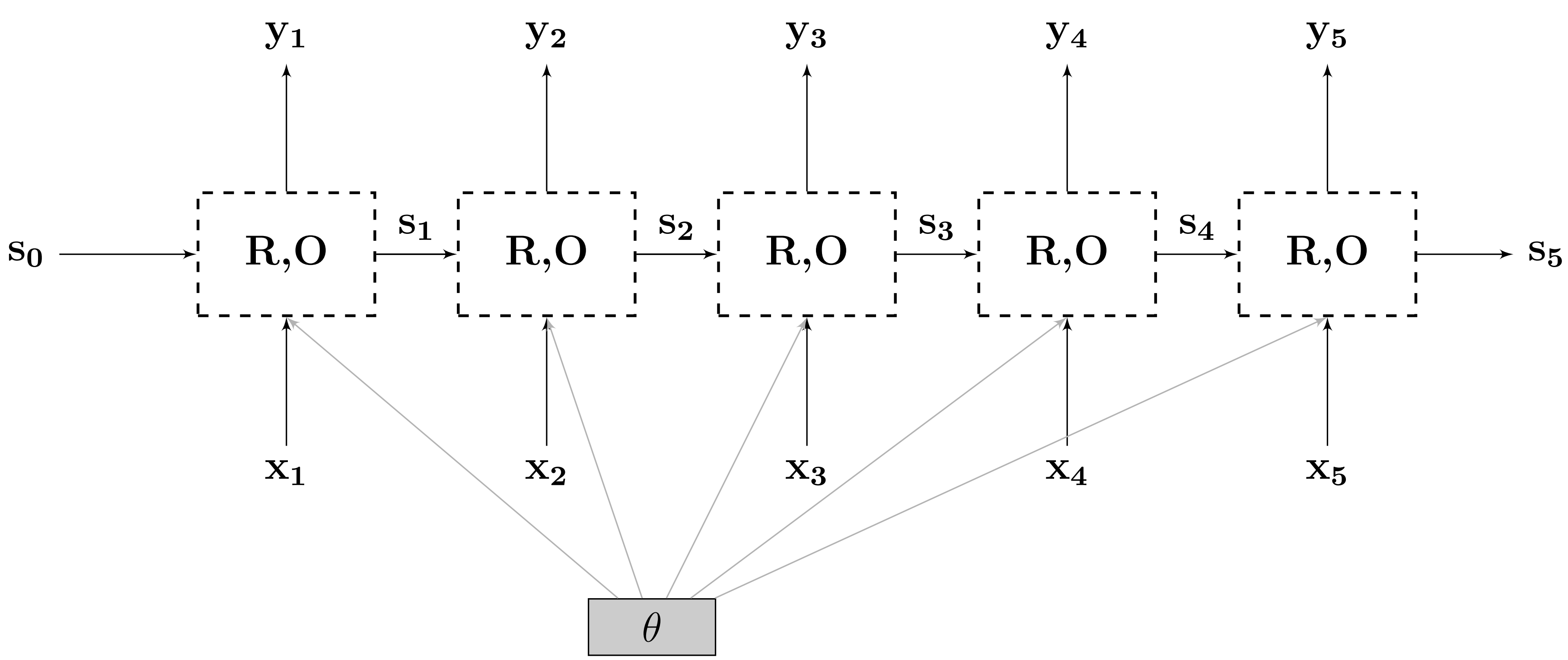
- What are the vectors $\mathbf{y_i}$ good for?



- On their own? **nothing**.

Recurrent Neural Networks

- What are the vectors $\mathbf{y_i}$ good for?

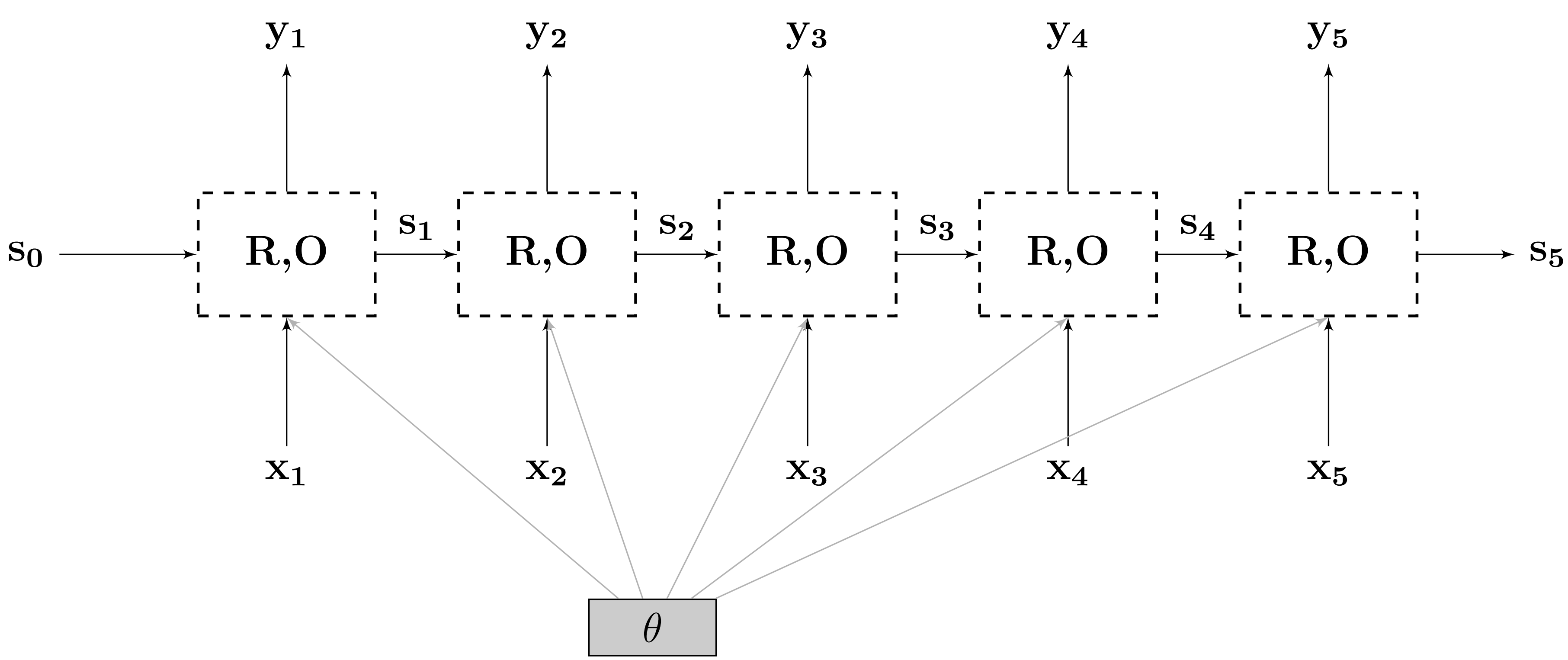


- On their own? **nothing**.

- **But we can train them.**

Recurrent Neural Networks

- What are the vectors $\mathbf{y_i}$ good for?



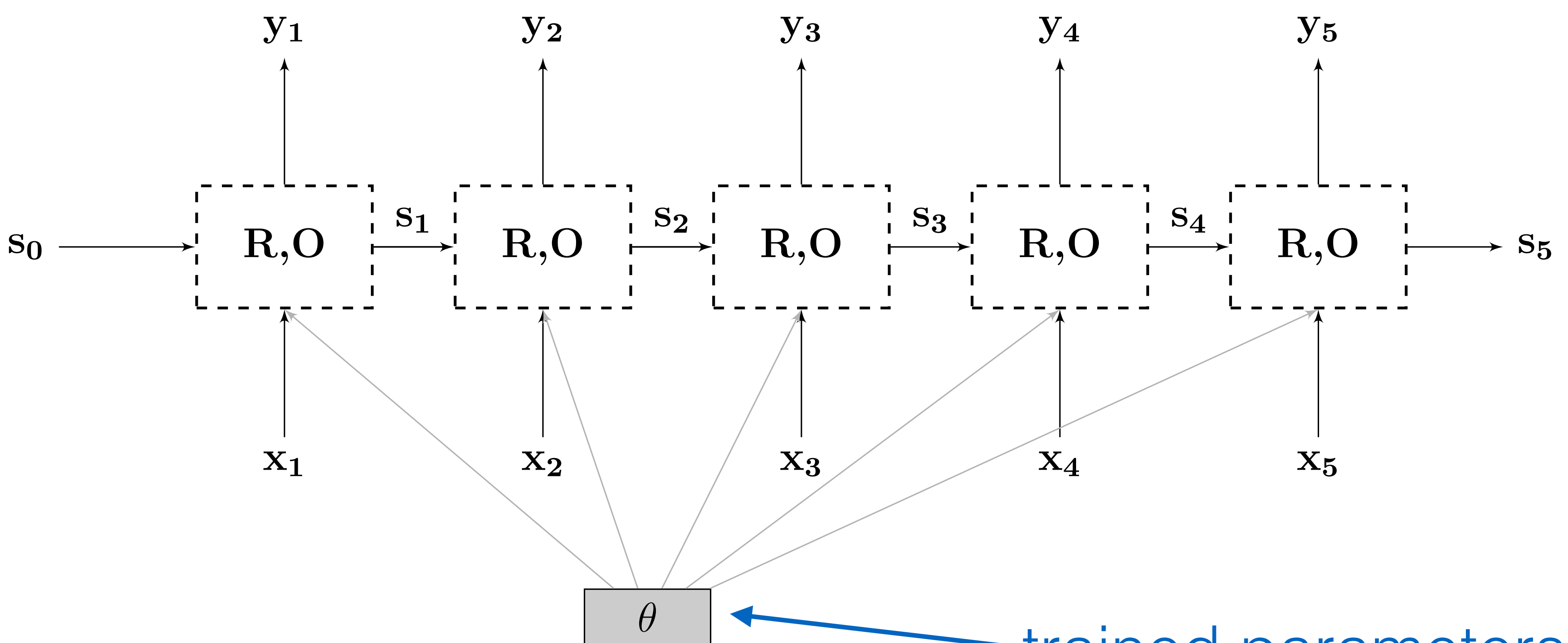
- On their own? **nothing**.
- **But we can train them.**

define function form

define loss

Recurrent Neural Networks

- What are the vectors $\mathbf{y_i}$ good for?



- On their own? **nothing**.

- **But we can train them.**

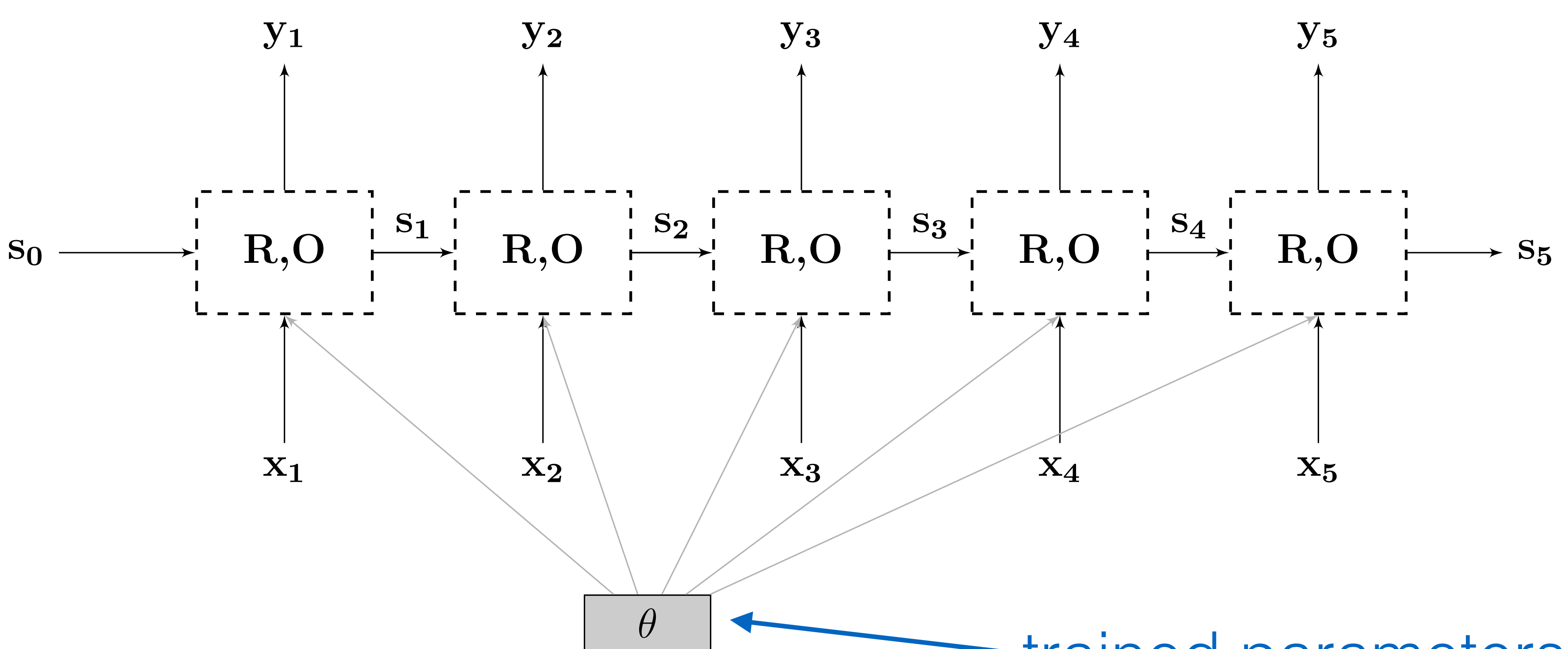
define function form

define loss

SimpleRNN:

$$R_{SRNN}(\mathbf{s_{i-1}}, \mathbf{x_i}) = tanh(\mathbf{W^s} \cdot \mathbf{s_{i-1}} + \mathbf{W^x} \cdot \mathbf{x_i})$$

looks simple.
theoretically powerful.
practically, not so much.



trained parameters.

- On their own? **nothing**.

- **But we can train them.**

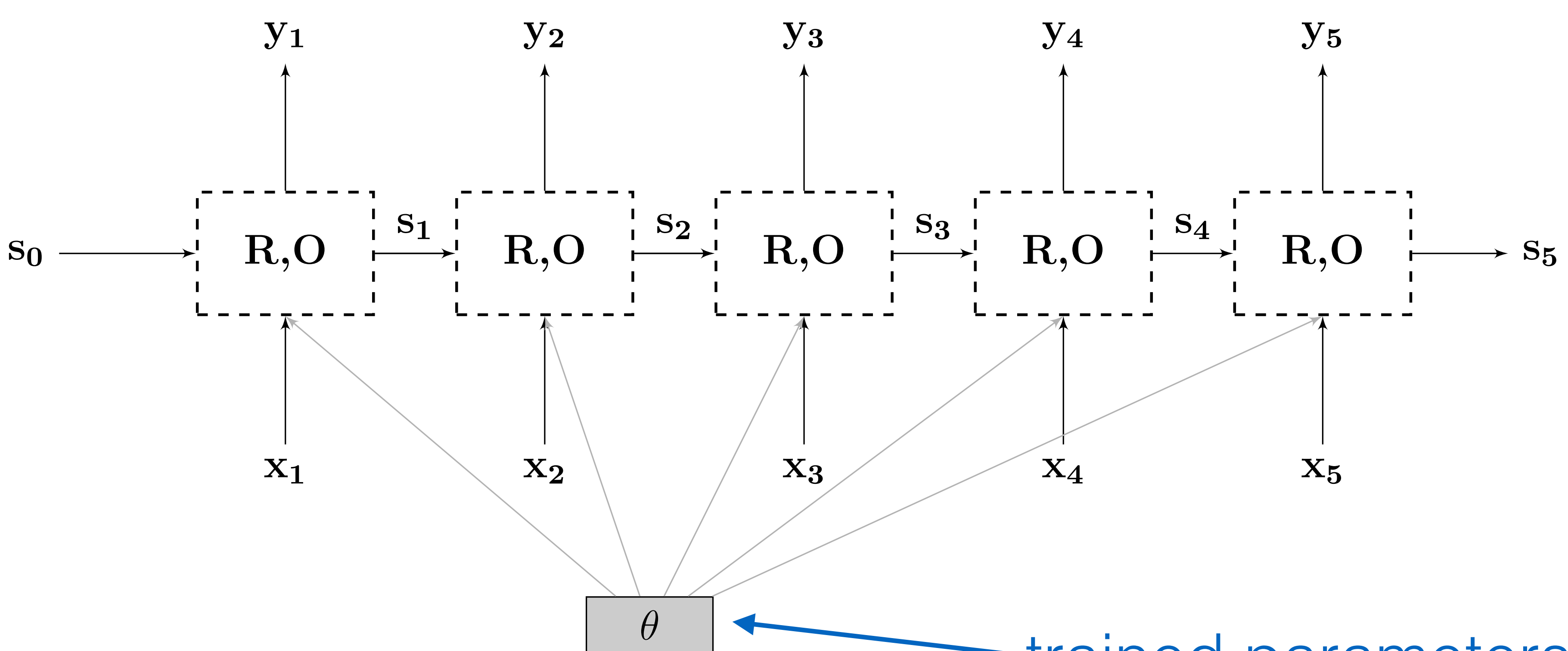
define function form

define loss

LSTM:

$$
\begin{aligned}
R_{LSTM}(\mathbf{s_{j-1}}, \mathbf{x_j}) =& [\mathbf{c_j}; \mathbf{h_j}] \\
\mathbf{c_j} =& \mathbf{c_{j-1}} \odot \mathbf{f} + \mathbf{g} \odot \mathbf{i} \\
\mathbf{h_j} =& \tanh(\mathbf{c_j}) \odot \mathbf{o} \\
\mathbf{i} =& \sigma(\mathbf{W^{xi}} \cdot \mathbf{x_j} + \mathbf{W^{hi}} \cdot \mathbf{h_{j-1}}) \\
\mathbf{f} =& \sigma(\mathbf{W^{xf}} \cdot \mathbf{x_j} + \mathbf{W^{hf}} \cdot \mathbf{h_{j-1}}) \\
\mathbf{o} =& \sigma(\mathbf{W^{xo}} \cdot \mathbf{x_j} + \mathbf{W^{ho}} \cdot \mathbf{h_{j-1}}) \\
\mathbf{g} =& \tanh(\mathbf{W^{xg}} \cdot \mathbf{x_j} + \mathbf{W^{hg}} \cdot \mathbf{h_{j-1}})
\end{aligned}
$$

looks complex, and is.
very strong in practice.



trained parameters.

- On their own? **nothing**.

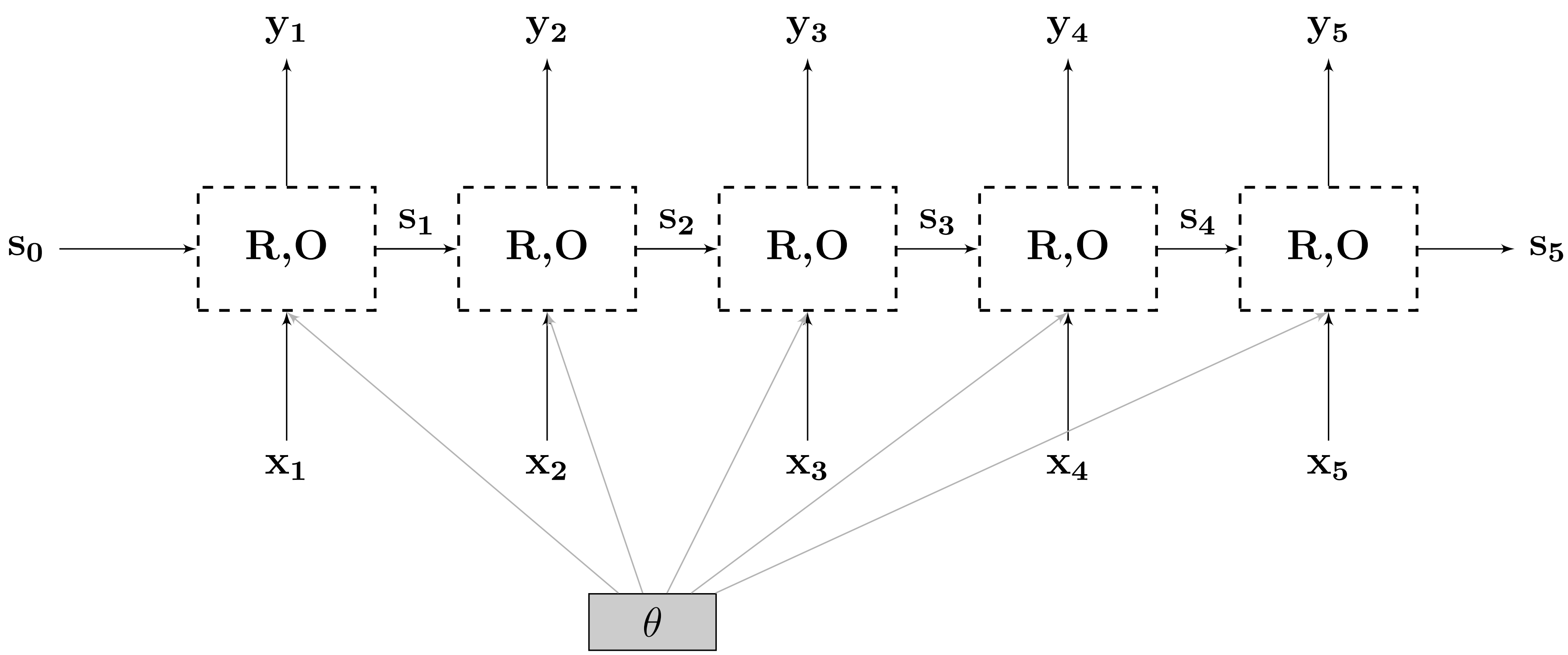
- **But we can train them.**

define function form

define loss

Recurrent Neural Networks

- What are the vectors $\mathbf{y_i}$ good for?



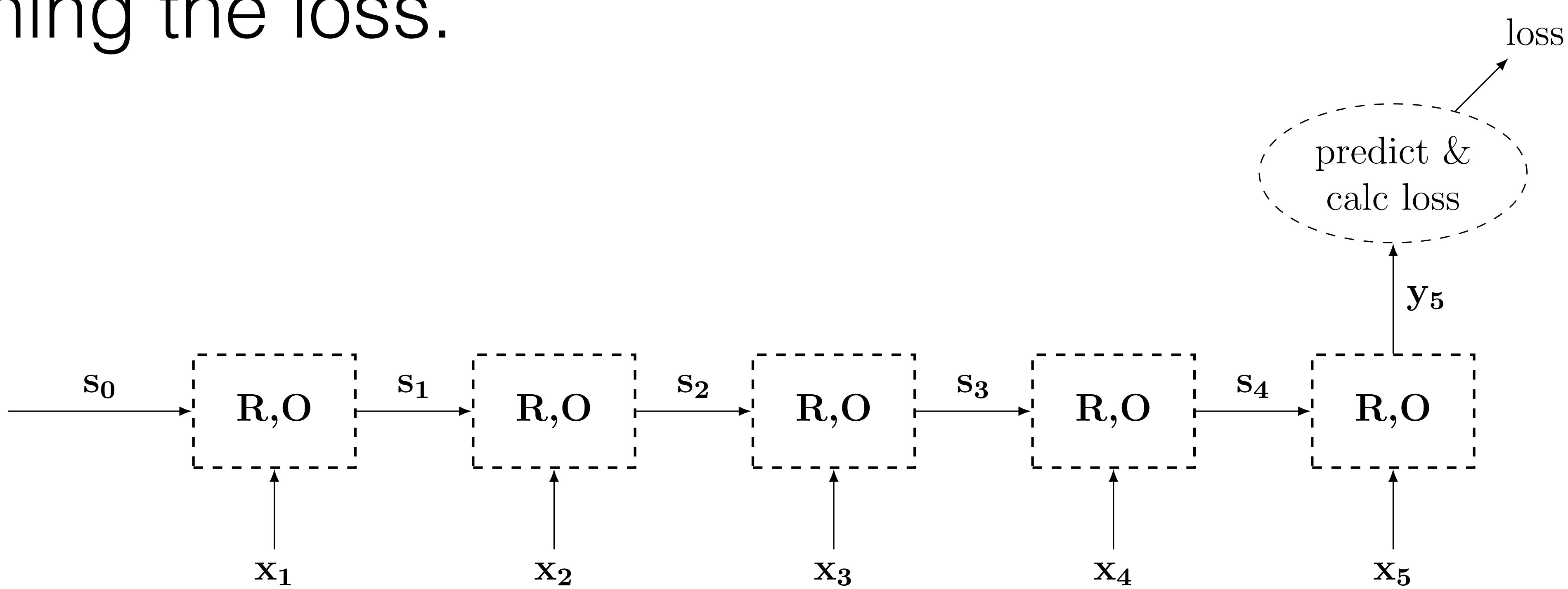
- On their own? **nothing**.
- **But we can train them.**

define function form

define loss

Recurrent Neural Networks

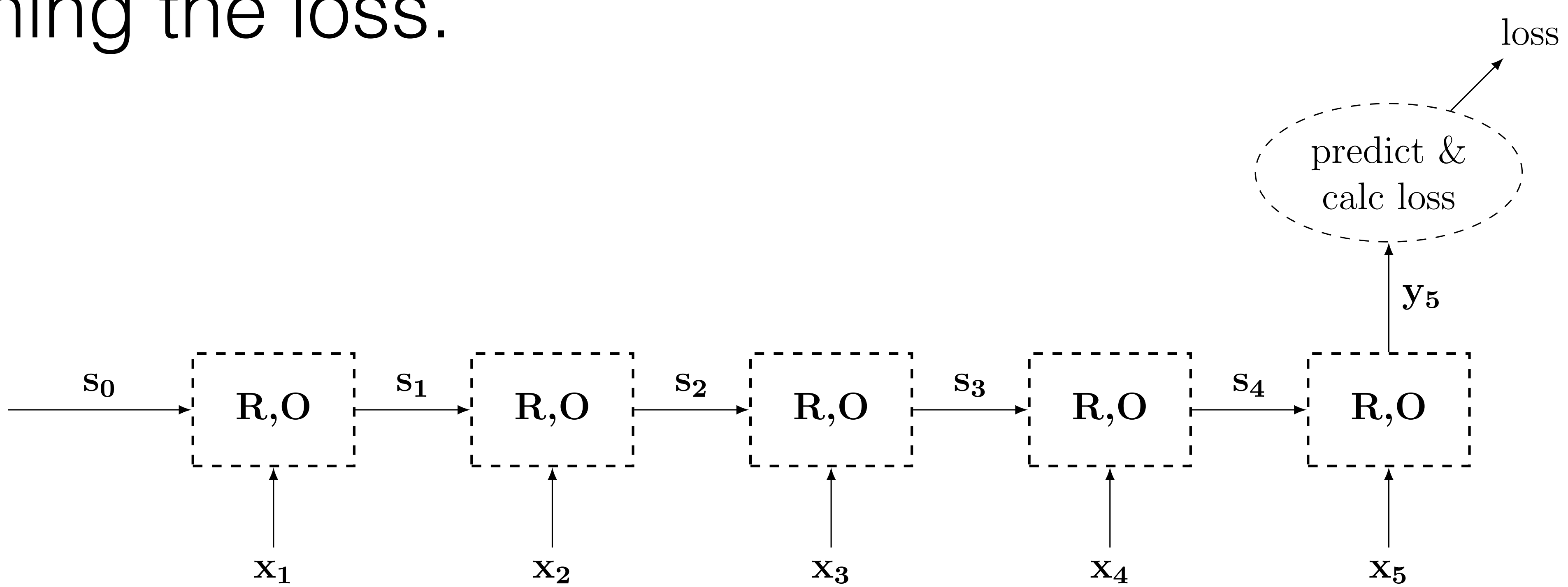
Defining the loss.



Acceptor: predict something from end state.
Backprop the error all the way back.
Train the network to capture meaningful information

Recurrent Neural Networks

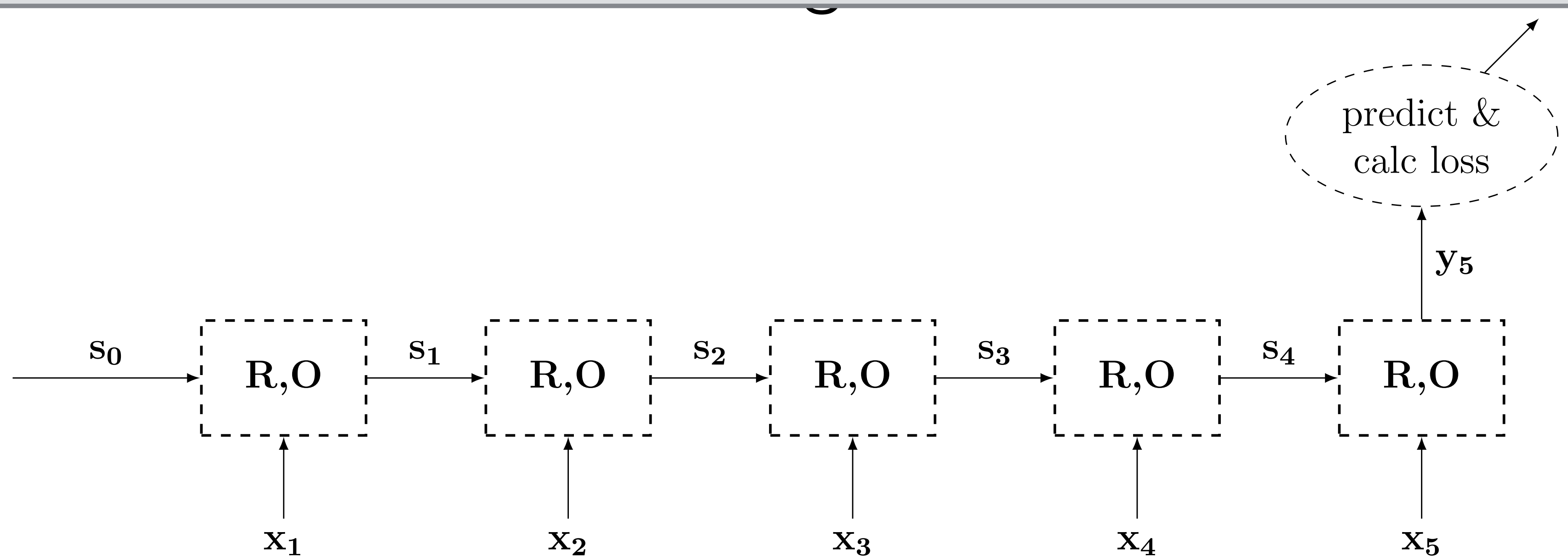
Defining the loss.



the final vector is a good "summary" of the sequence

Acceptor: predict something from end state.
Backprop the error all the way back.
Train the network to capture meaningful information

- Predict sentiment of the sentence based on all words.
- Predict word i based on words 1,...,i-1.

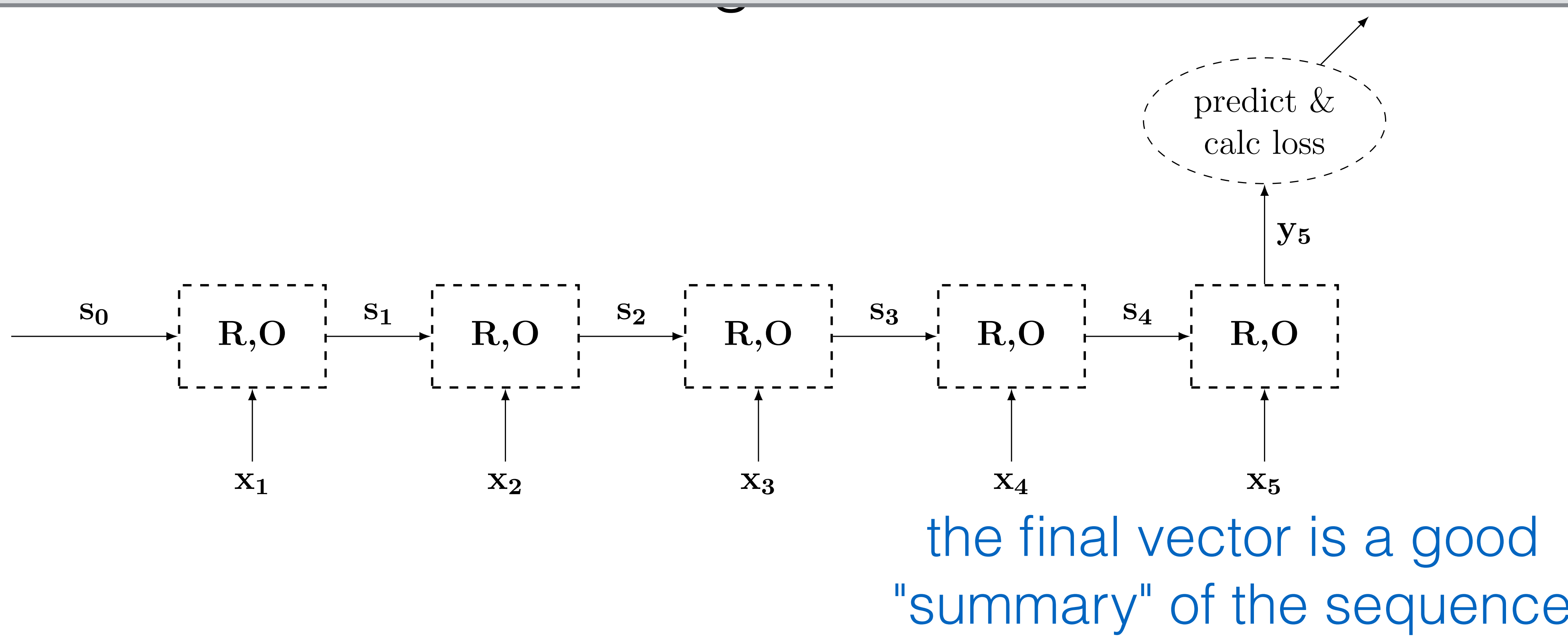


the final vector is a good "summary" of the sequence

Acceptor: predict something from end state.
Backprop the error all the way back.
Train the network to capture meaningful information

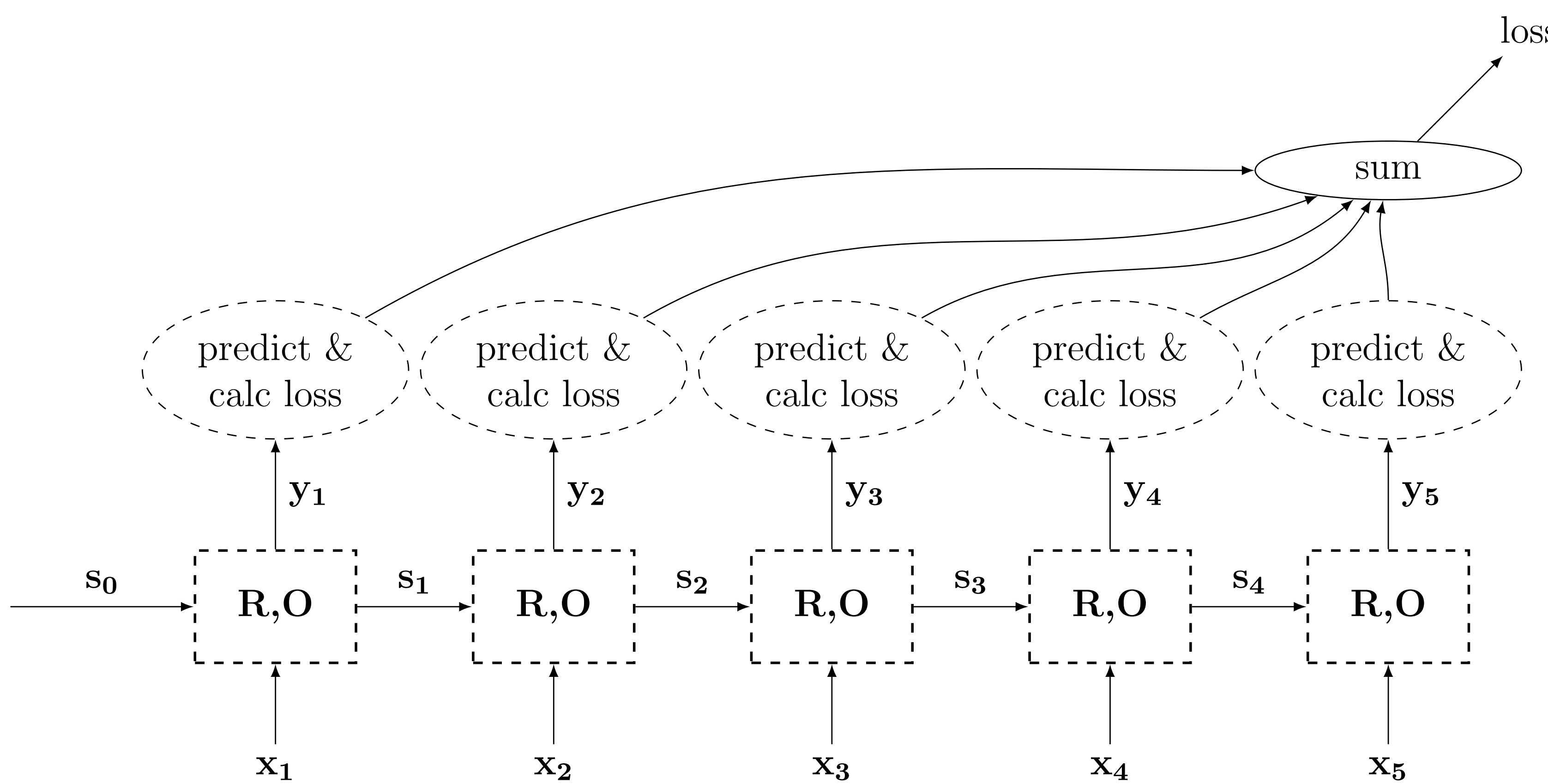
- Predict sentiment of the sentence based on all words.

- **Predict word i based on words 1,...,i-1.**



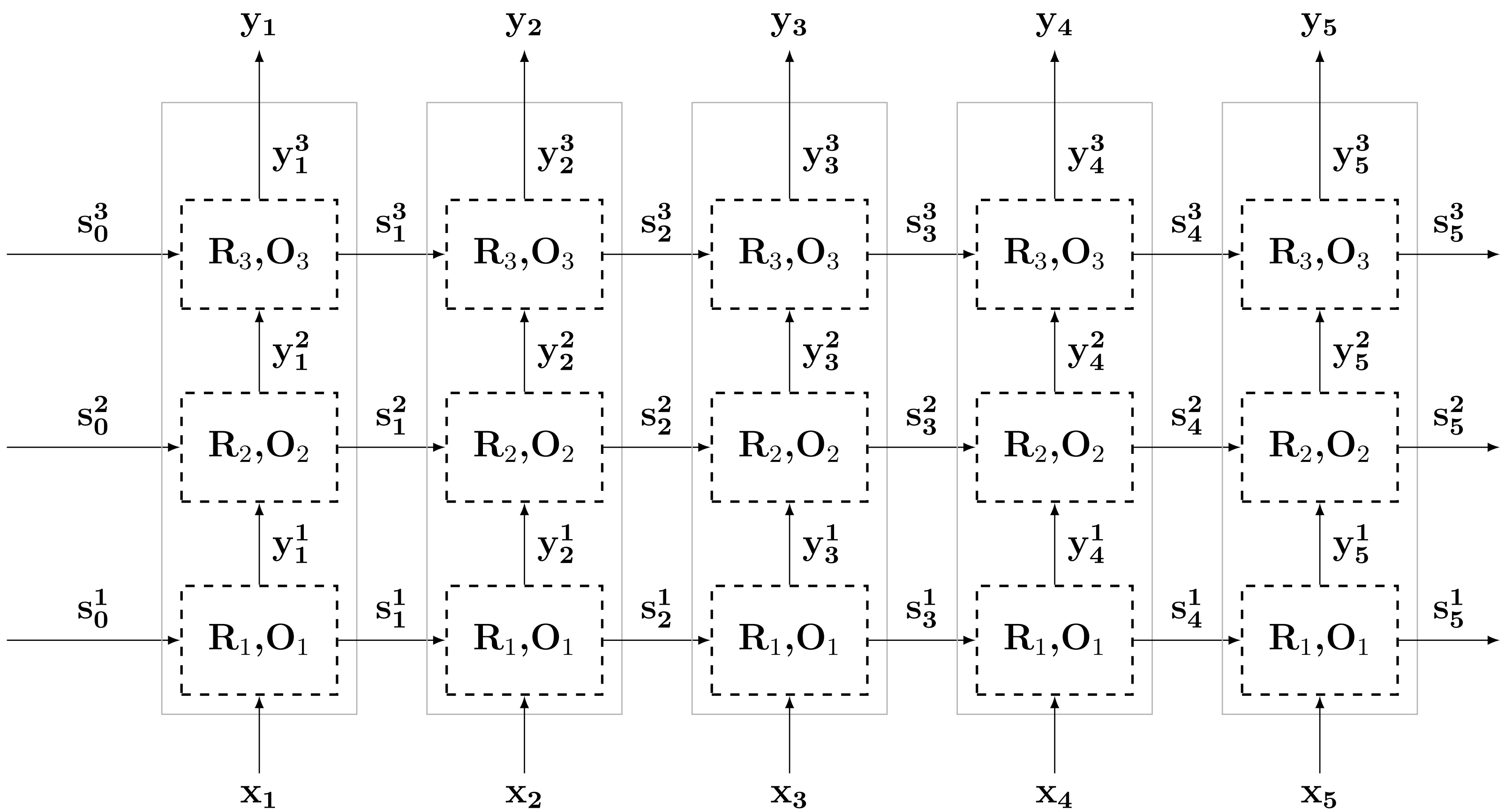
Acceptor: predict something from end state.
Backprop the error all the way back.
Train the network to capture meaningful information

Recurrent Neural Networks



Transducer: predict something from each state.
Backprop the sum of errors all the way back.
Train the network to capture meaningful information

"Deep RNNs"



RNN can be stacked
deeper is better!
(better how?)

Story so far:

- There is a thing called a (deep) RNN.
- We can feed it a list of vectors.
    - Each vector represents a word.
- At the end it spits out a vector summarizing the list of vectors.
- We influence the summarization with training.

Story so far:

Sequence in, vector out.

But human language is not (only) a sequence!

- At the end it spits out a vector summarizing the list of vectors.
- We influence the summarization with training.

Can RNNs learn hierarchy?

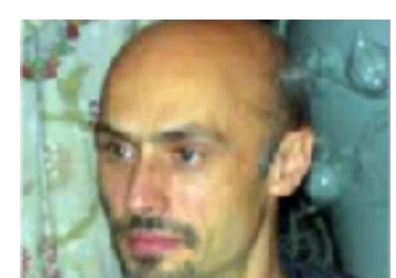
Can RNNs learn hierarchy?

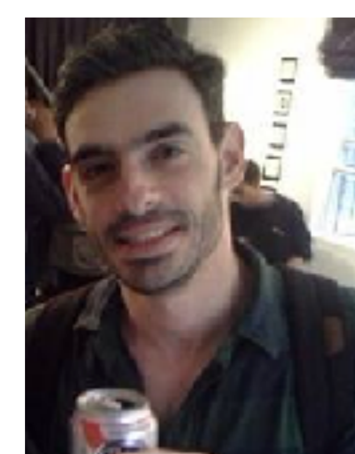
(joint work with Tal Linzen and Emmanuel Dupoux)

Assessing the Ability of LSTMs to Learn Syntax-Sensitive Dependencies

Tal Linzen[1,2] **Emmanuel Dupoux**[1]
LSCP[1] & IJN[2], CNRS,
EHESS and ENS, PSL Research University
{tal.linzen,
emmanuel.dupoux}@ens.fr

Yoav Goldberg
Computer Science Department
Bar Ilan University
yoav.goldberg@gmail.com





The case for Syntax

- Some natural-language phenomena are indicative of hierarchical structure.

- For example, subject verb agreement.

the boy kicks the ball

the boys kick the ball

The case for Syntax

- Some natural-language phenomena are indicative of hierarchical structure.

- For example, subject verb agreement.

the boy with the white shirt with the blue collar kicks the ball

the boys with the white shirts with the blue collars kick the ball

The case for Syntax

- Some natural-language phenomena are indicative of hierarchical structure.
- For example, subject verb agreement.

the boy (with the white shirt (with the blue collar)) kicks the ball

he boys (with the white shirts (with the blue collars)) kick the ball

The case for Syntax

- Some natural-language phenomena are indicative of hierarchical structure.

- For example, subject verb agreement.

the boy (with the white shirt (with the blue collar)) kicks the ball

he boys (with the white shirts (with the blue collars)) kick the ball

nsubj

Can a sequence LSTM learn agreement?

Can a sequence LSTM learn agreement?

some prominent figures in the history of philosophy who have defended moral rationalism are plato and immanuel kant .

Can a sequence LSTM learn agreement?

some prominent figures in the history of philosophy who have defended moral NN   are plato and immanuel kant .

replace rare words with their POS

Can a sequence LSTM learn agreement?

some prominent figures in the history of philosophy who have defended moral NN are plato and immanuel kant .

choose a verb with a subject

Can a sequence LSTM learn agreement?

some prominent figures in the history of philosophy who have defended moral NN ____

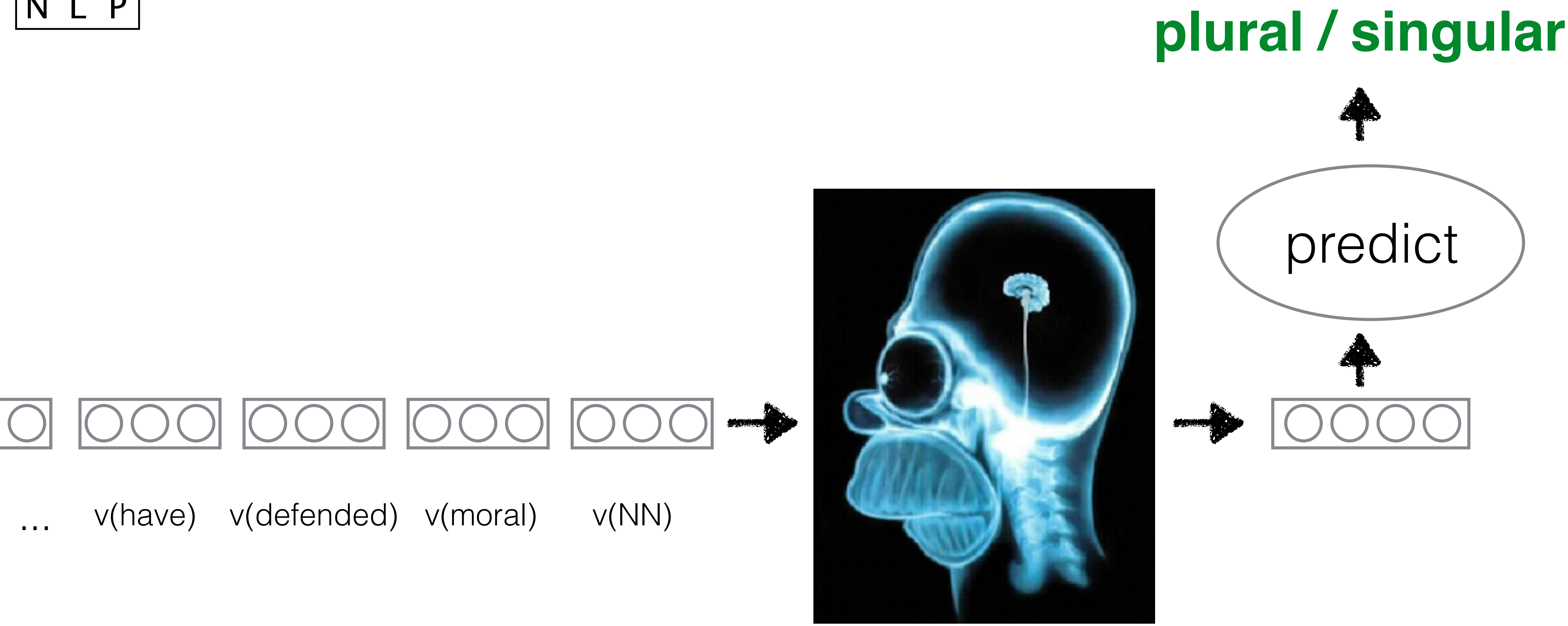
cut the sentence at the verb

Can a sequence LSTM learn agreement?

some prominent figures in the history of philosophy who have defended moral NN ____

plural or singular?

binary prediction task


plural / singular
predict
...
v(have)
v(defended)
v(moral)
v(NN)

Can a sequence LSTM learn agreement?

some prominent figures in the history of philosophy who have defended moral NN ____

plural or singular?

binary prediction task

Can a sequence LSTM learn agreement?

some prominent figures in the history of philosophy who have defended moral NN ____

plural or singular?

Can a sequence LSTM learn agreement?

some prominent figures in the history of philosophy who have defended moral NN ____

plural or singular?

Can a sequence LSTM learn agreement?

some prominent figures in the history of philosophy who have defended moral NN ____

plural or singular?

Can a sequence LSTM learn agreement?

some prominent figures in the history of philosophy who have defended moral NN ____

plural or singular?

in order to answer:

Need to learn the concept of number.

Need to identify the **subject** (ignoring irrelevant words)

Somewhat Harder Task

Somewhat Harder Task

some prominent figures in the history of philosophy who have defended moral NN are plato and immanuel kant .

choose a verb with a subject

Somewhat Harder Task

some prominent figures in the history of philosophy who have defended moral NN are plato and immanuel kant .

some prominent figures in the history of philosophy who have defended moral NN is plato and immanuel kant .

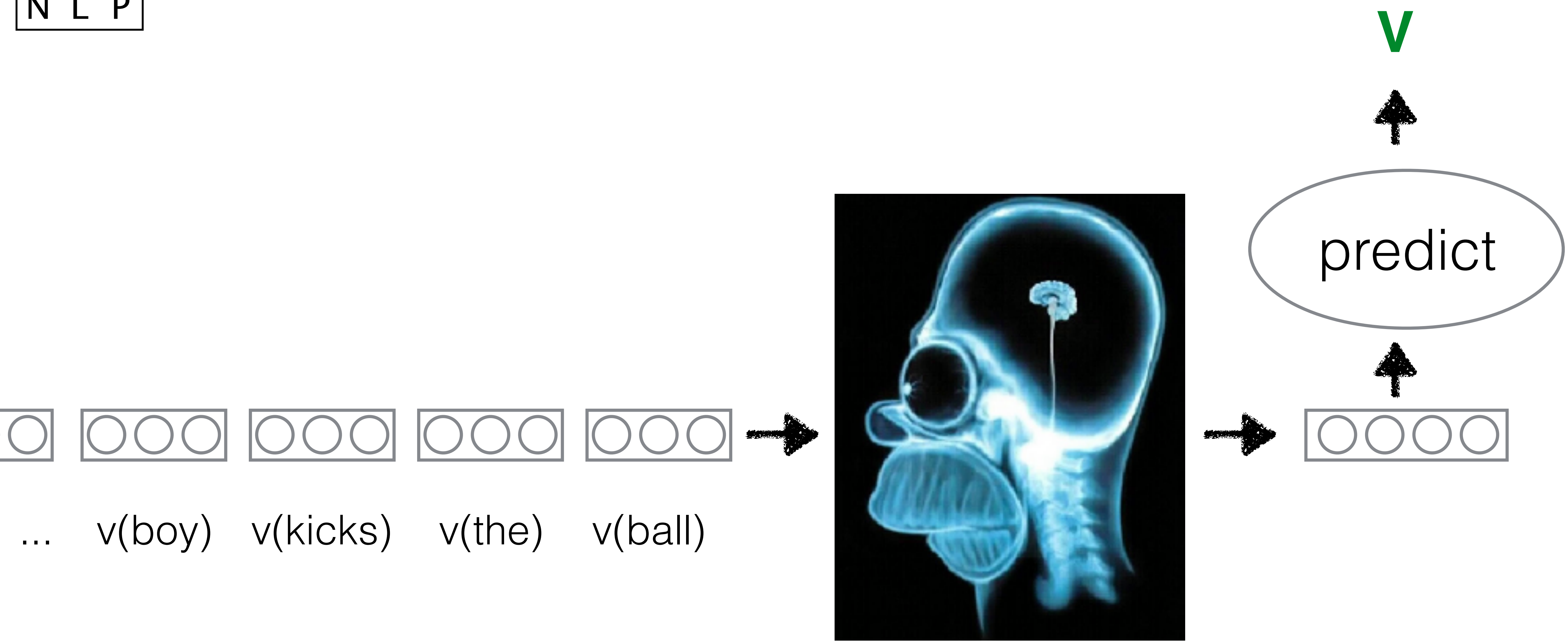
choose a verb with a subject
and flip its number.

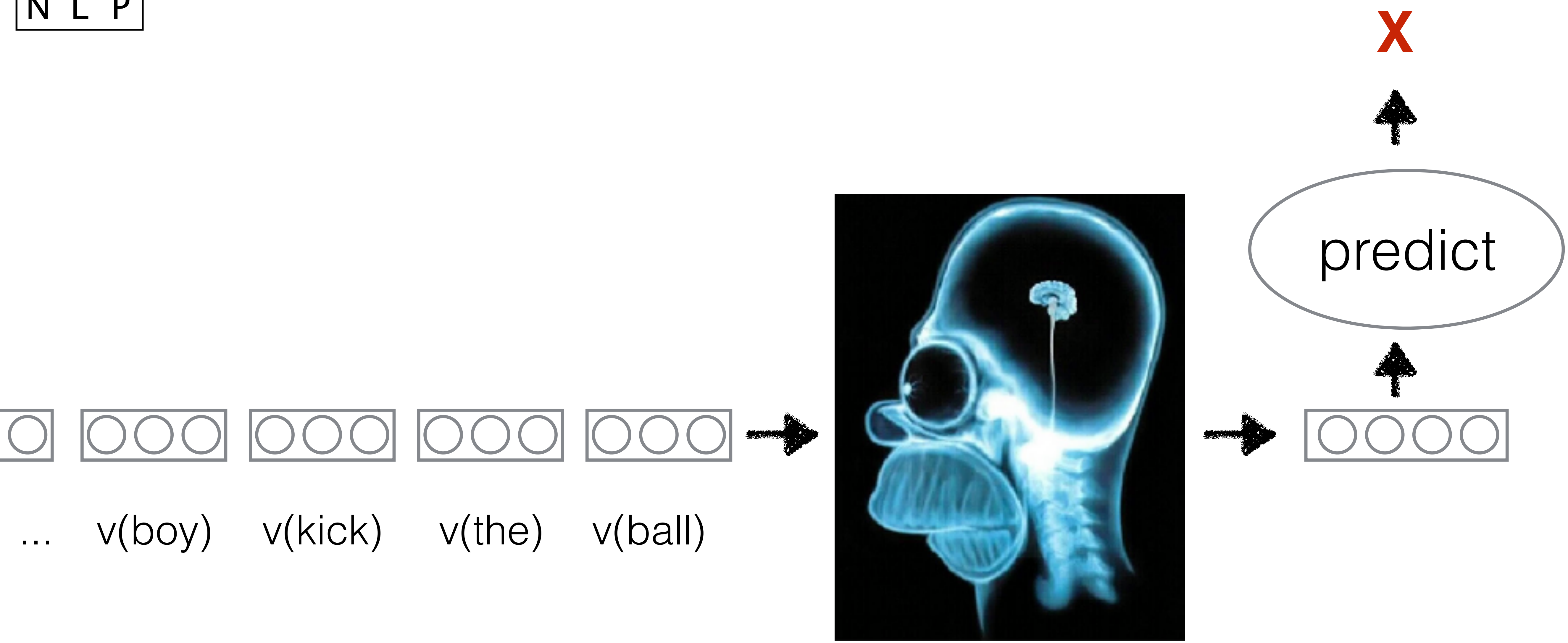
Somewhat Harder Task

some prominent figures in the history of philosophy who have defended moral NN are plato and immanuel kant . V

some prominent figures in the history of philosophy who have defended moral NN is plato and immanuel kant . X

can the LSTM learn to distinguish good from bad sentences?


V
predict
...
v(boy)
v(kicks)
v(the)
v(ball)


x
predict
...
v(boy)
v(kick)
v(the)
v(ball)

Can a sequence LSTM learn agreement?

LSTMs learn agreement remarkably well.

predicts number with **99**% accuracy.

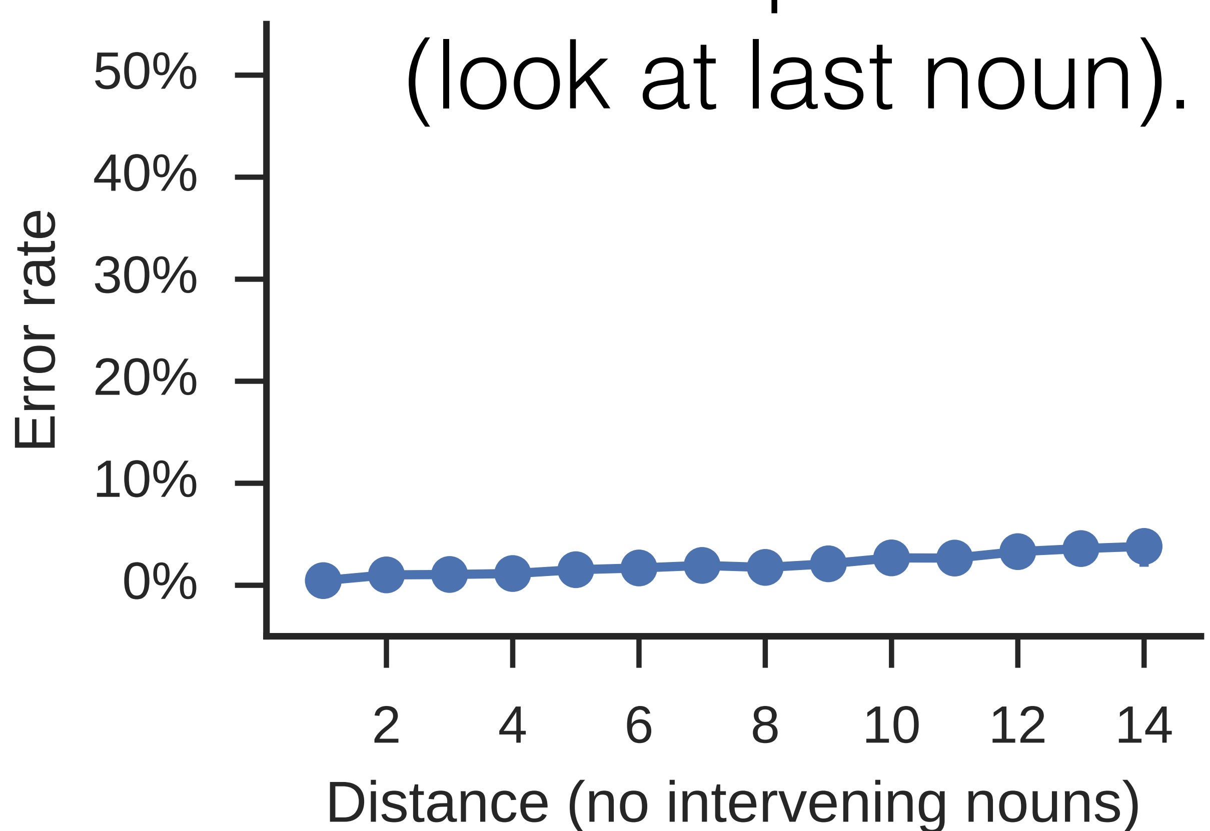
...but most examples are very easy
(look at last noun).

Can a sequence LSTM learn agreement?

LSTMs learn agreement remarkably well.

predicts number with **99**% accuracy.

...but most examples are very easy (look at last noun).



Can a sequence LSTM learn agreement?

LSTMs learn agreement remarkably well.

predicts number with **99**% accuracy.

...but most examples are very easy
(look at last noun).

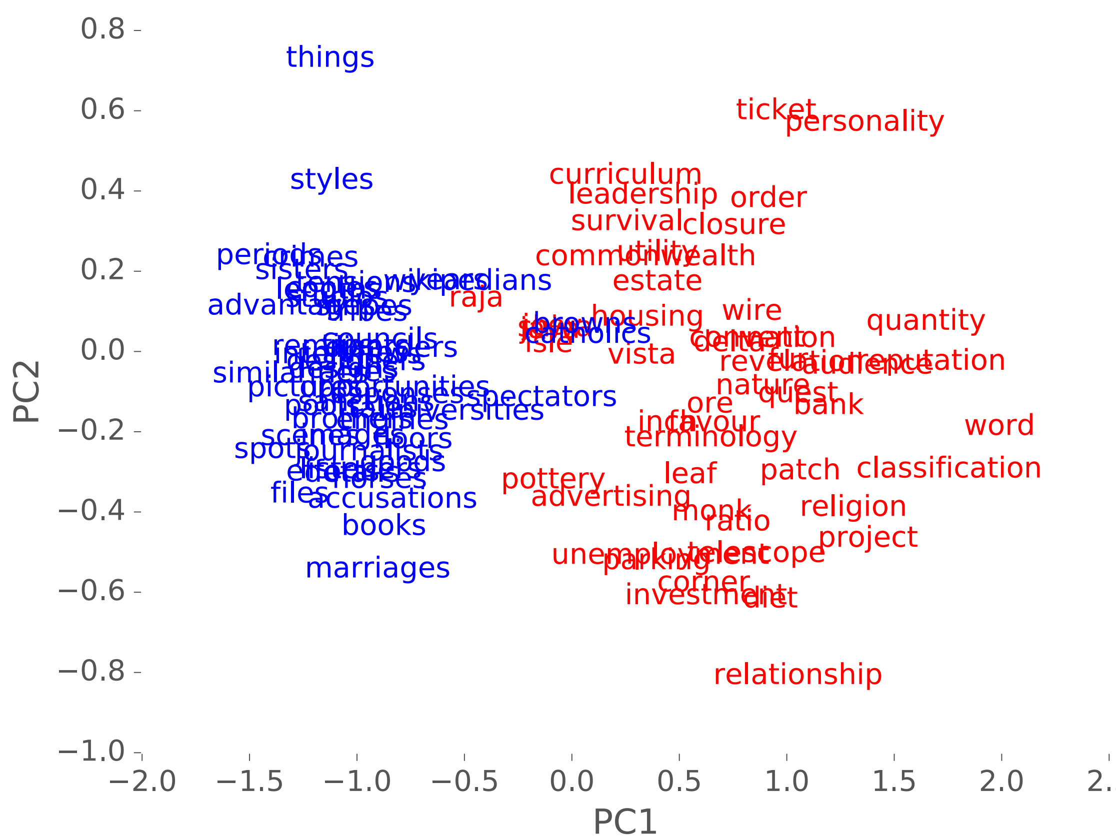
when restricted to cases
of at least one intervening noun:

97% accuracy

Can a sequence LSTM learn agreement?

LSTMs learn agreement remarkably well.

learns number of nouns

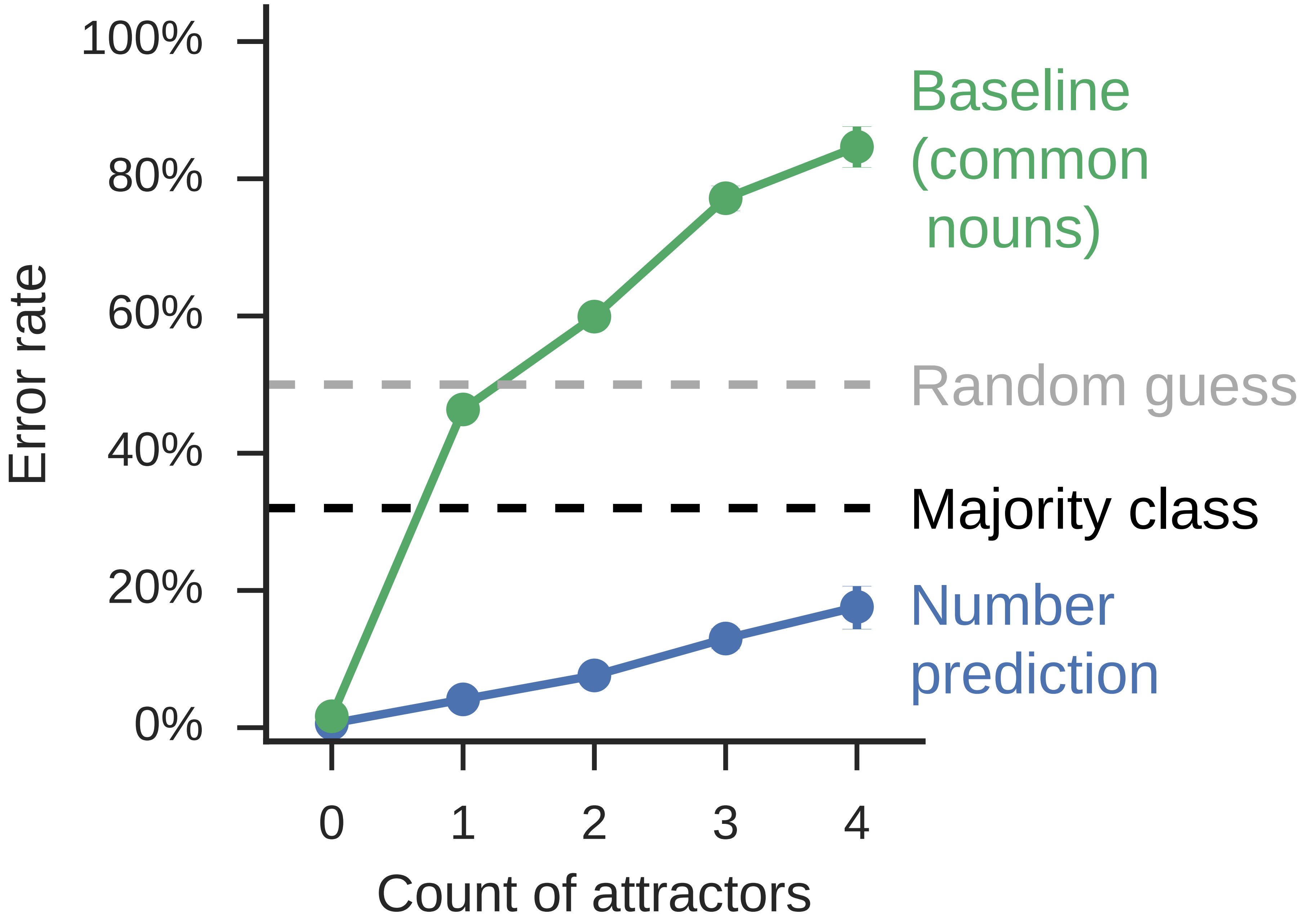


Can a sequence LSTM learn agreement?

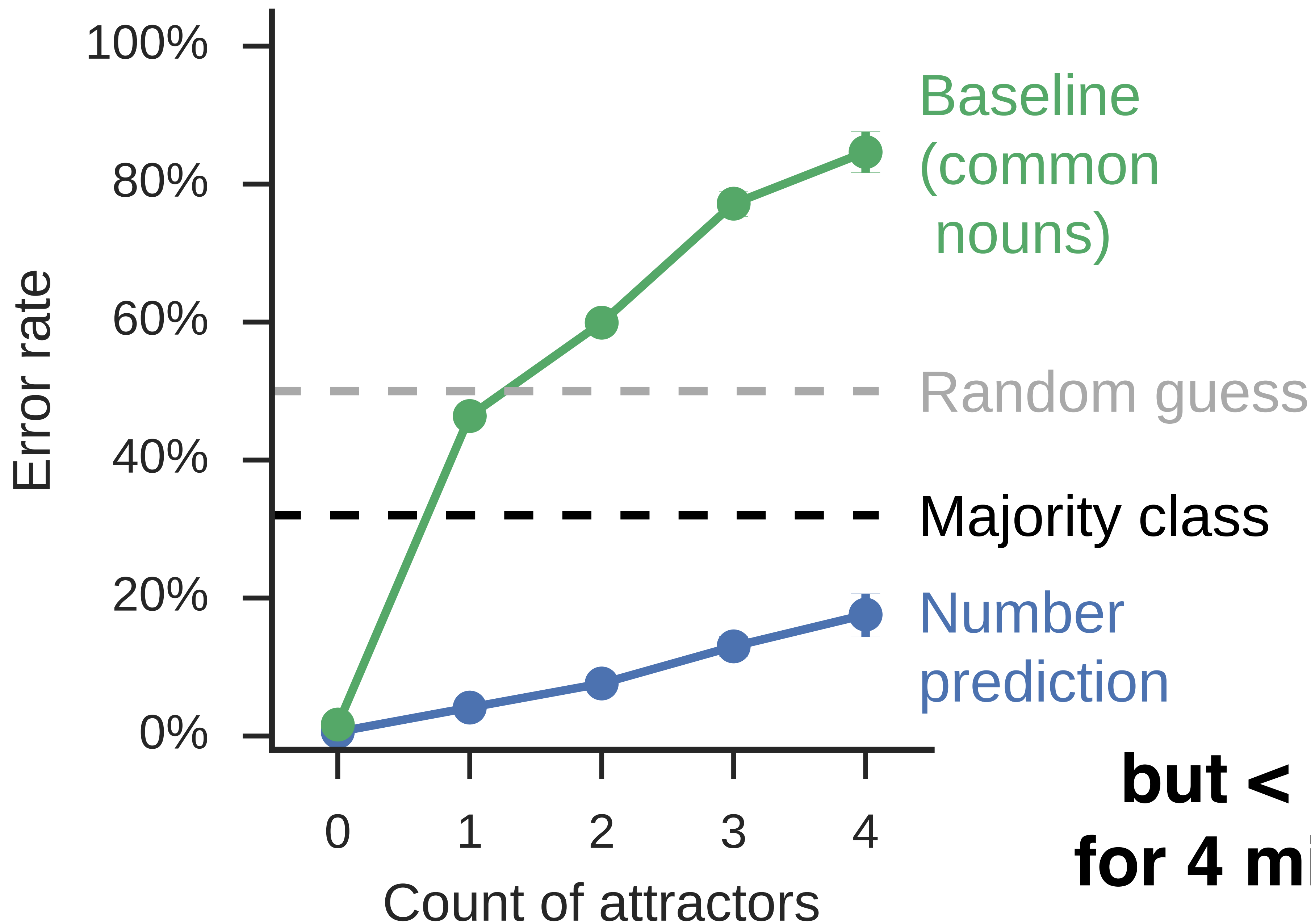
LSTMs learn agreement remarkably well.

more errors as the number of intervening nouns of opposite number increases

Can a sequence LSTM learn agreement?



Can a sequence LSTM learn agreement?



but < 16% err for 4 misleading nouns...

Can a sequence LSTM learn agreement?

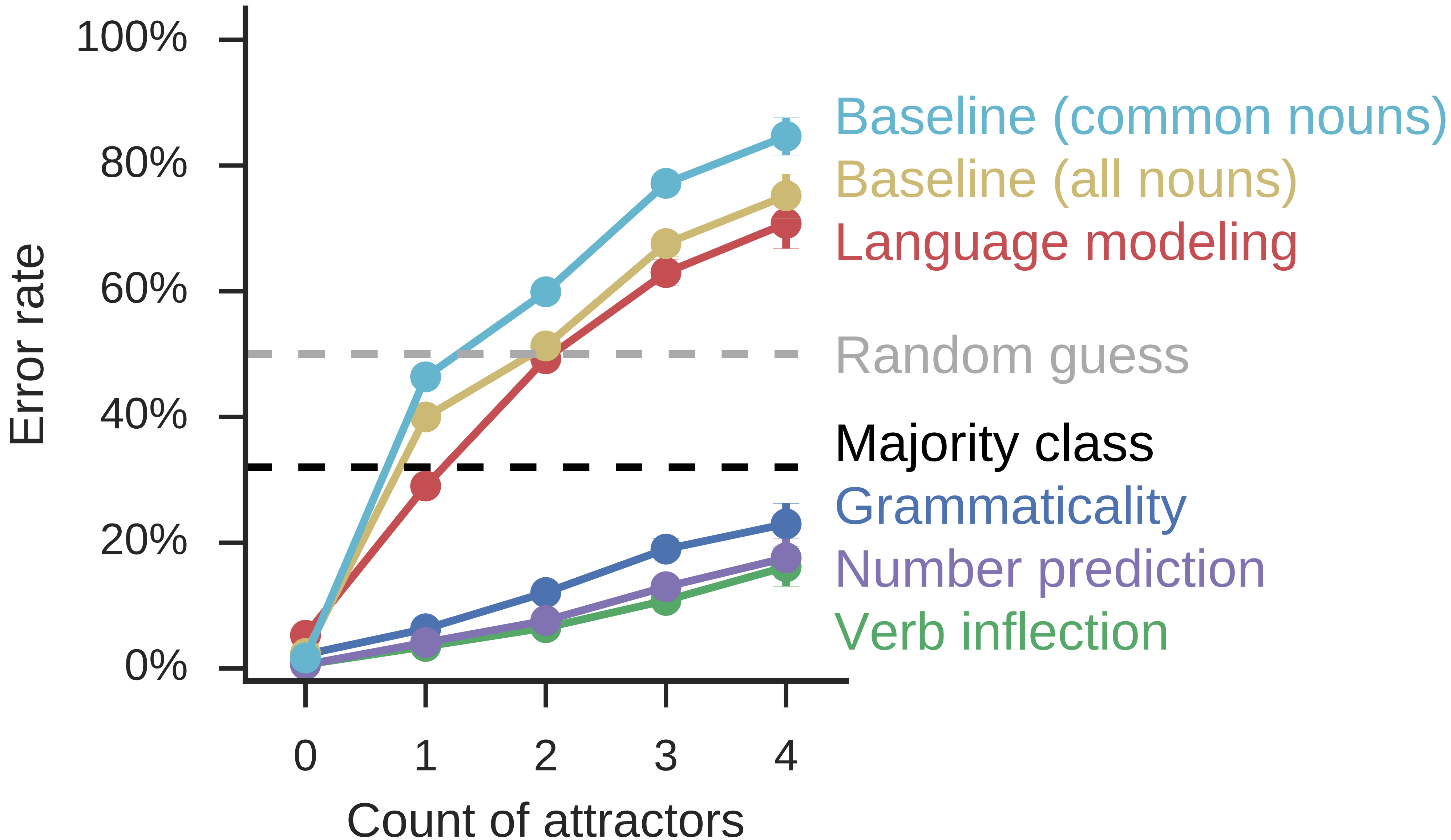
LSTMs learn agreement remarkably well.

but we trained it on the agreement task.

does a language model learn agreement?

Can a sequence LSTM learn agreement?

does a language model learn agreement?



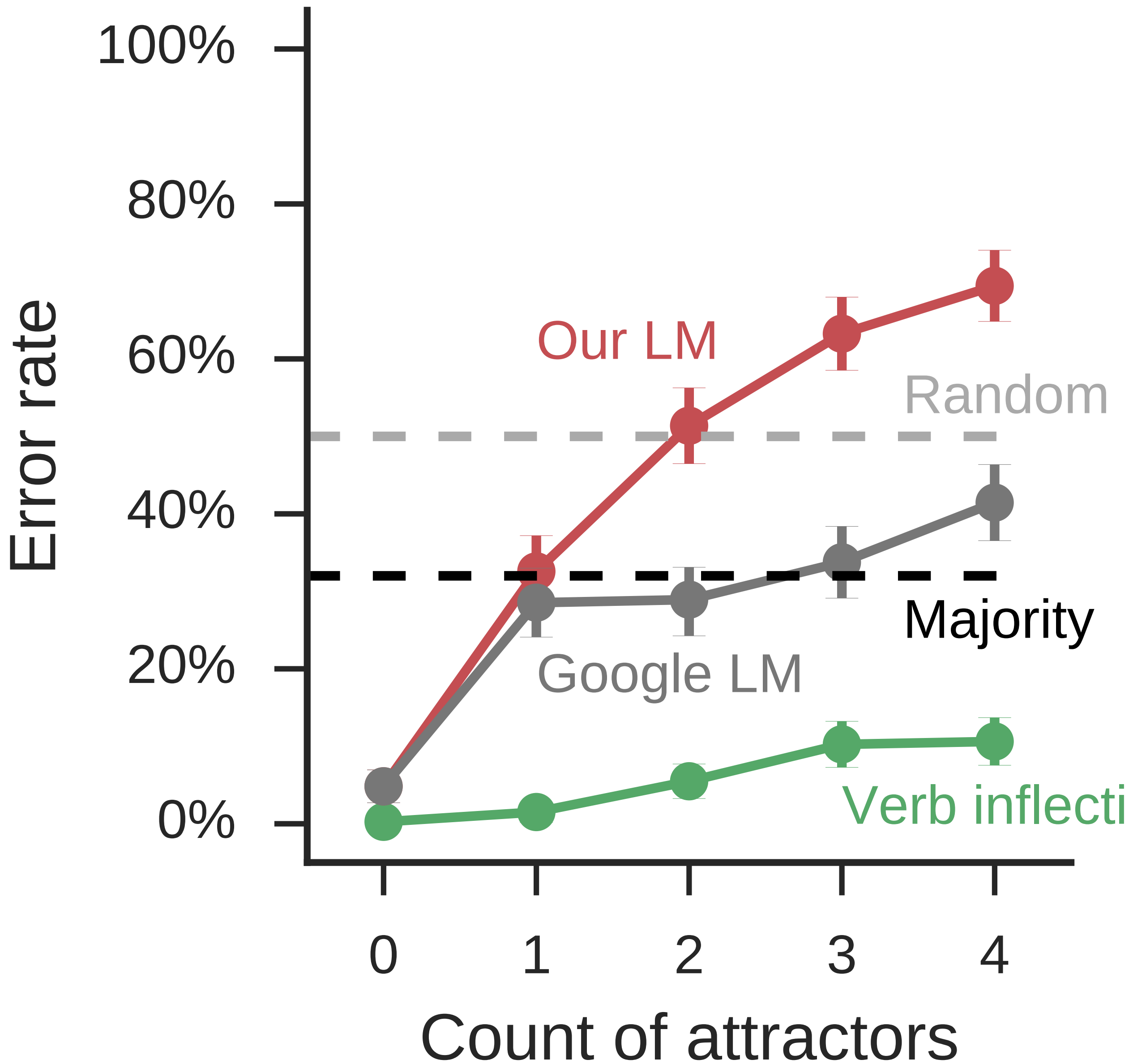
Can a sequence LSTM learn agreement?

does a language model learn agreement?

what if we used the **best LM in the world?**

Can a sequence LSTM learn agreement?

does a language model learn agreement?



Google's beast LM does better than ours but still struggles considerably.

Can a sequence LSTM learn agreement?

does a language model learn agreement?

LSTM-LM **does not** learn agreement.

Explicit error signal is required.

but with explicit signal,
LSTMs can learn agreement very well.

Can a sequence LSTM learn agreement?

Where do LSTMs fail?

in many and diverse cases.

but we did manage to find some common trends.

Can a sequence LSTM learn agreement?

Where do LSTMs fail?

noun compounds can be tricky

Conservation **refugees live** in a world colored in shades of gray; limbo.

Can a sequence LSTM learn agreement?

Where do LSTMs fail?

Relative clauses are hard.

The **landmarks** *that* this article lists here **are** also run-of-the-mill and not notable.

Can a sequence LSTM learn agreement?

Where do LSTMs fail?

Reduced relative clauses are harder.

The **landmarks** this article lists here **are** also run-of-the-mill and not notable.

Can a sequence LSTM learn agreement?

Where do LSTMs fail?

	Error
No relative clause	3.2%
Overt relative clause	9.9%
Reduced Relative clause	**25%**

Can a sequence LSTM learn agreement?

Where do LSTMs fail?

	Error
No relative clause	3.2%
Overt relative clause	9.9%
Reduced Relative clause	**25%**

humans also fail much more on reduced relatives.

The agreement experiment: recap

- We wanted to show LSTMs can't learn hierarchy.
  - --> We sort-of failed.
- LSTMs learn to cope with natural-language patterns that exhibit hierarchy, based on minimal and indirect supervision.
- But some sort of relevant supervision is required.

What happens beyond English?

- English is a simple language.
- We started exploring more interesting ones.
- If you want to collaborate on cool agreement patterns in your favorite language, let's discuss!

Story so far:

- RNNs are very flexible sequence encoders.
- We can train them to encode rather intricate syntactic structures.

Story so far:

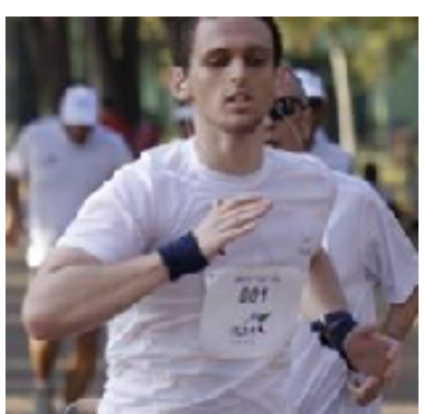
- RNNs are very flexible sequence encoders.
- We can train them to encode rather intricate syntactic structures.
- **Can we use them for parsing?**

Parsing with LSTMs, Take 1

Easy-First Dependency Parsing with Hierarchical Tree LSTMs

Eliyahu Kiperwasser
Computer Science Department
Bar-Ilan University
Ramat-Gan, Israel
elikip@gmail.com

Yoav Goldberg
Computer Science Department
Bar-Ilan University
Ramat-Gan, Israel
yoav.goldberg@gmail.com

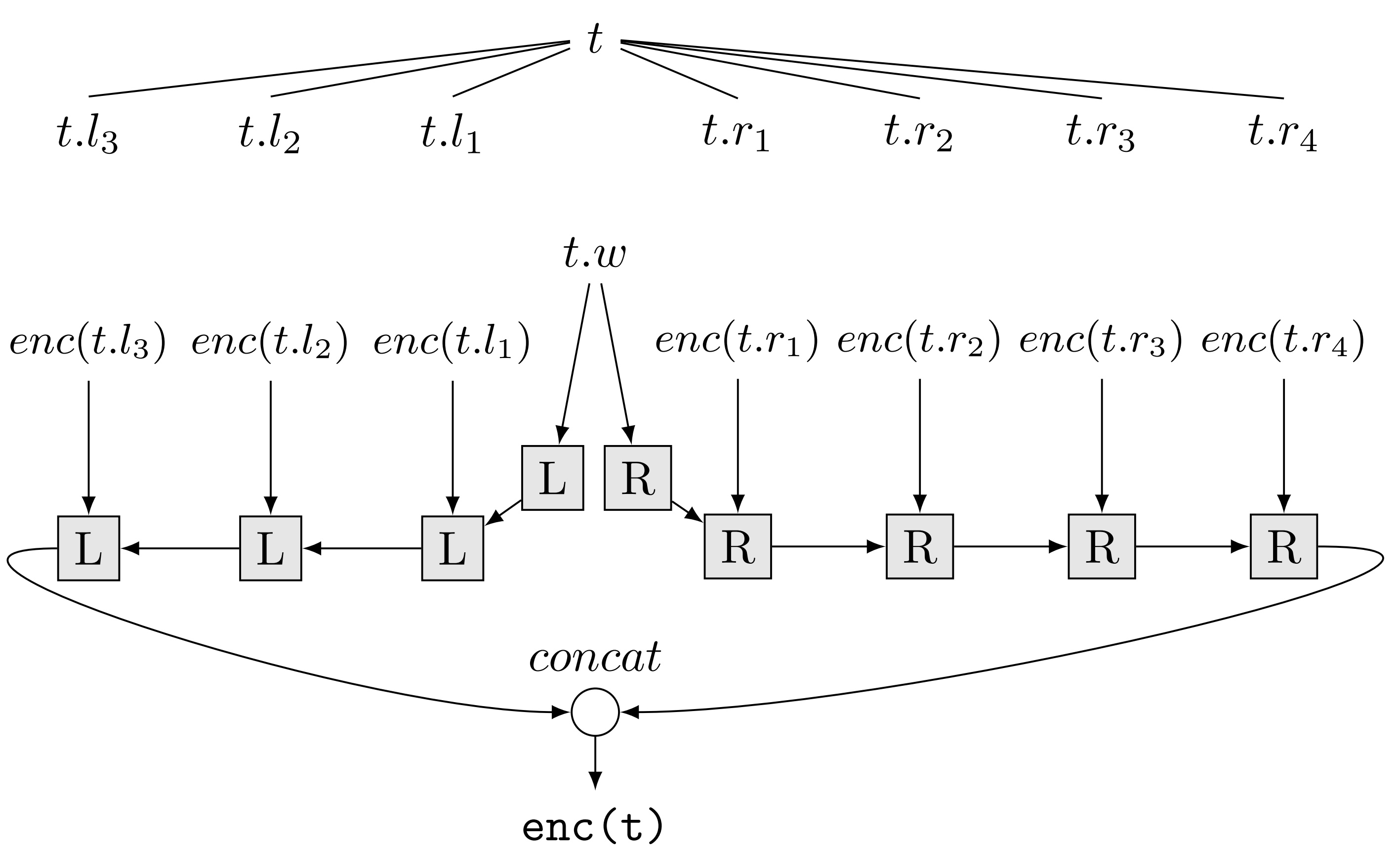


Core Idea

- LSTMs are SOTA at modeling sequences.
- Encode sequence of modifiers as an LSTM.
- Combine in a recursive manner.

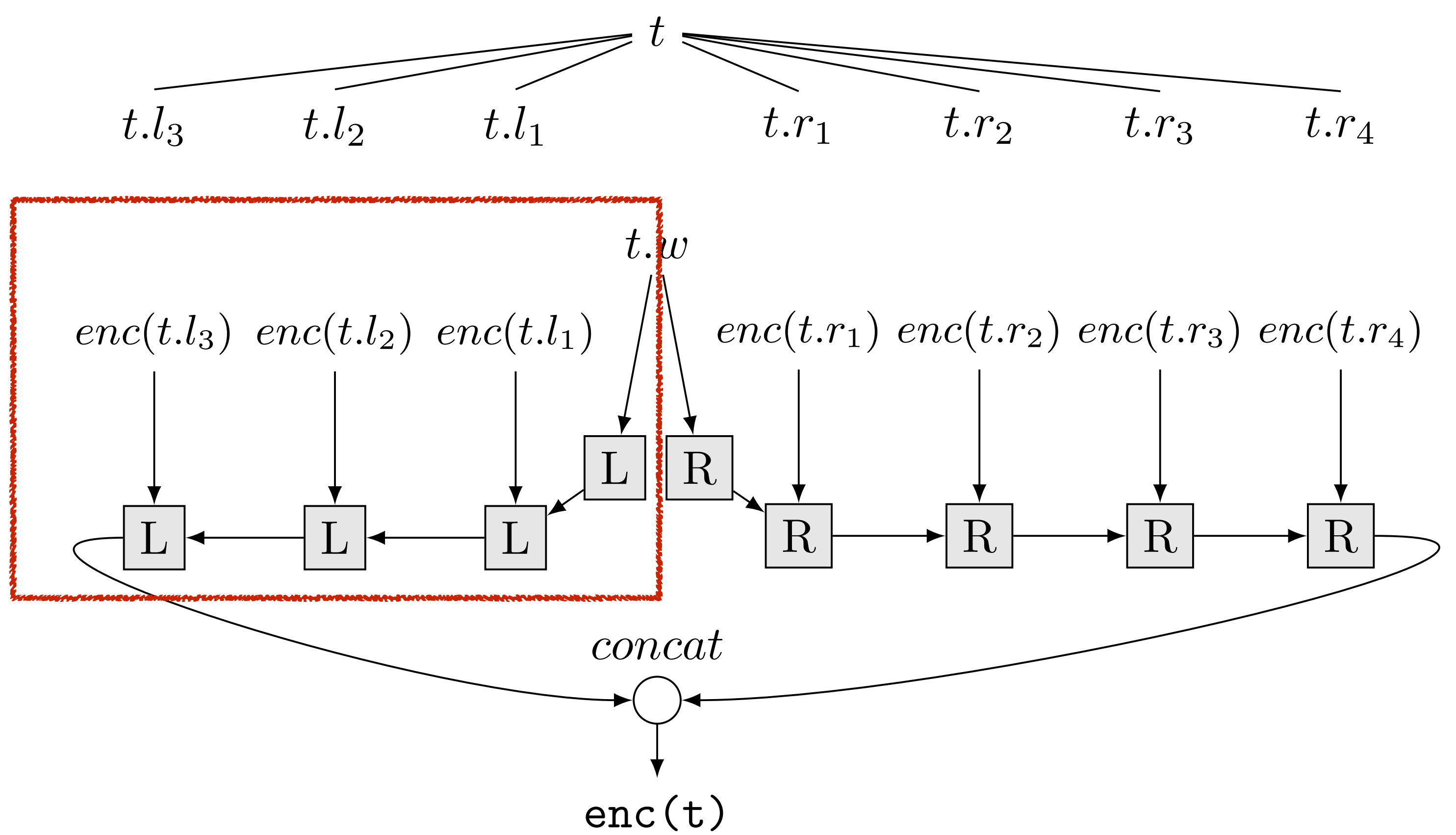
great for dependency trees.

Core Idea



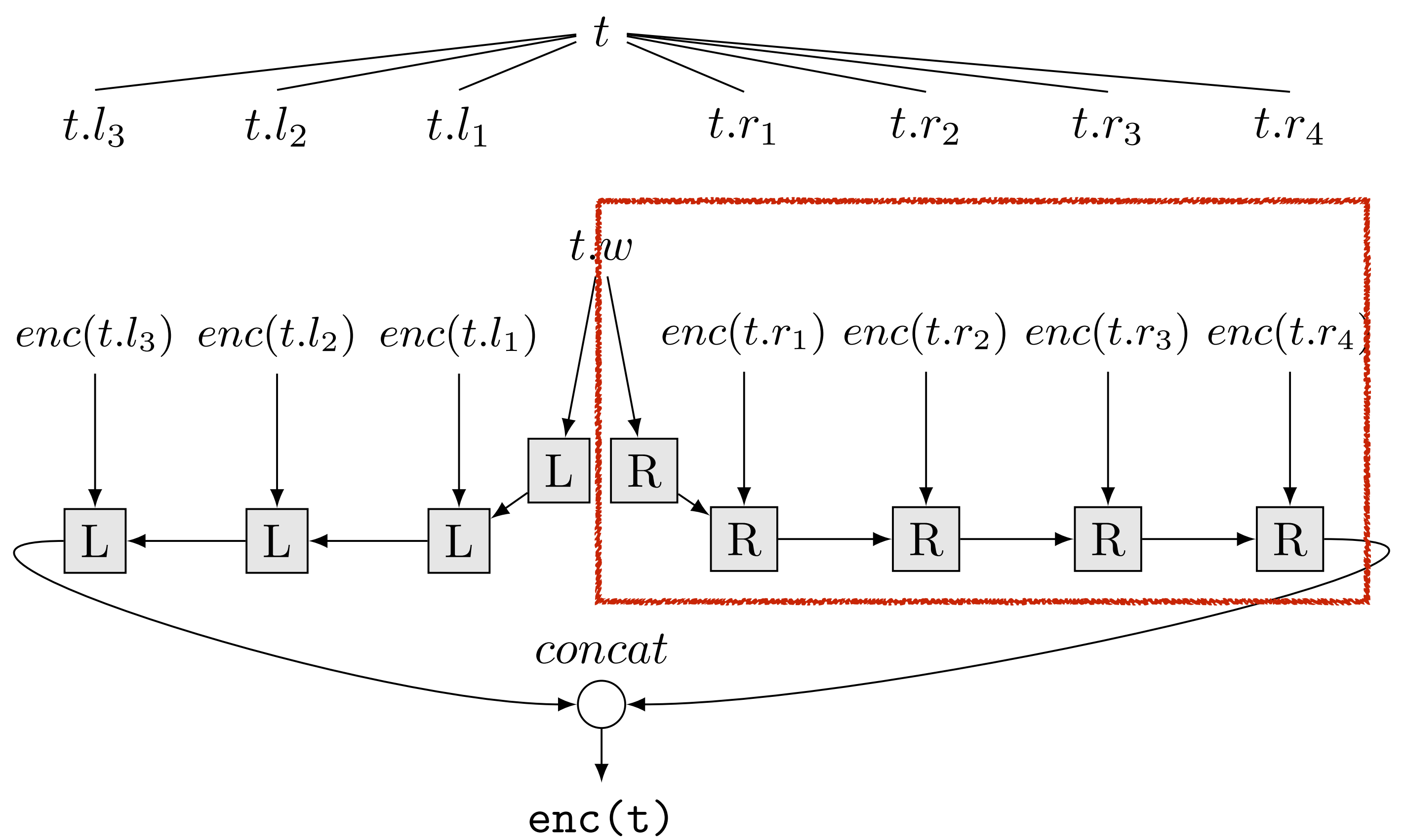
- Two LSTMs
- head + Left modifiers encoded w/ LSTM-L
- head + Right modifiers encoded w/ LSTM-R
- The Left and Right end states are concatenated

Core Idea



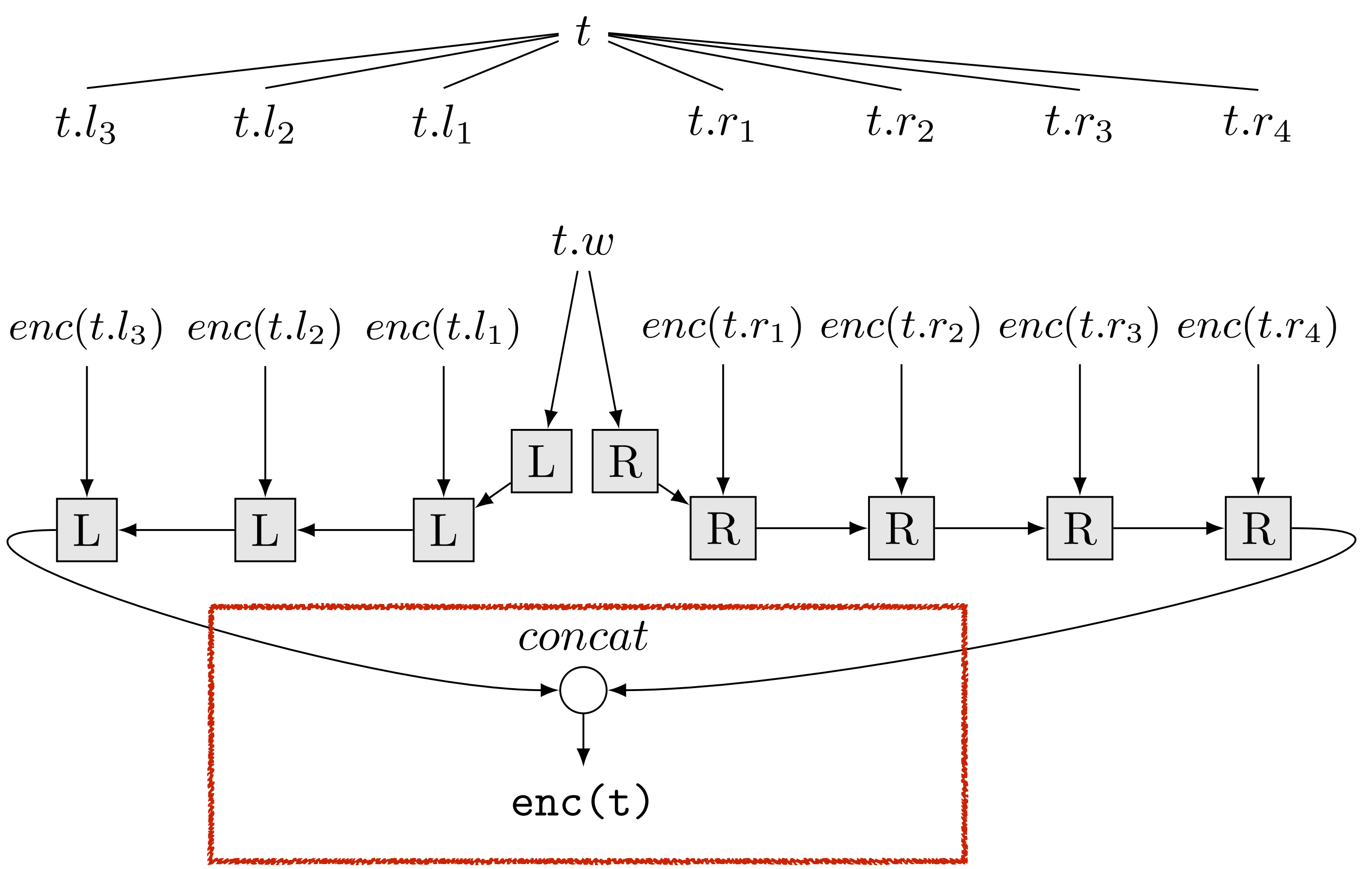
- Two LSTMs
- head + Left modifiers encoded w/ LSTM-L
- head + Right modifiers encoded w/ LSTM-R
- The Left and Right end states are concatenated

Core Idea

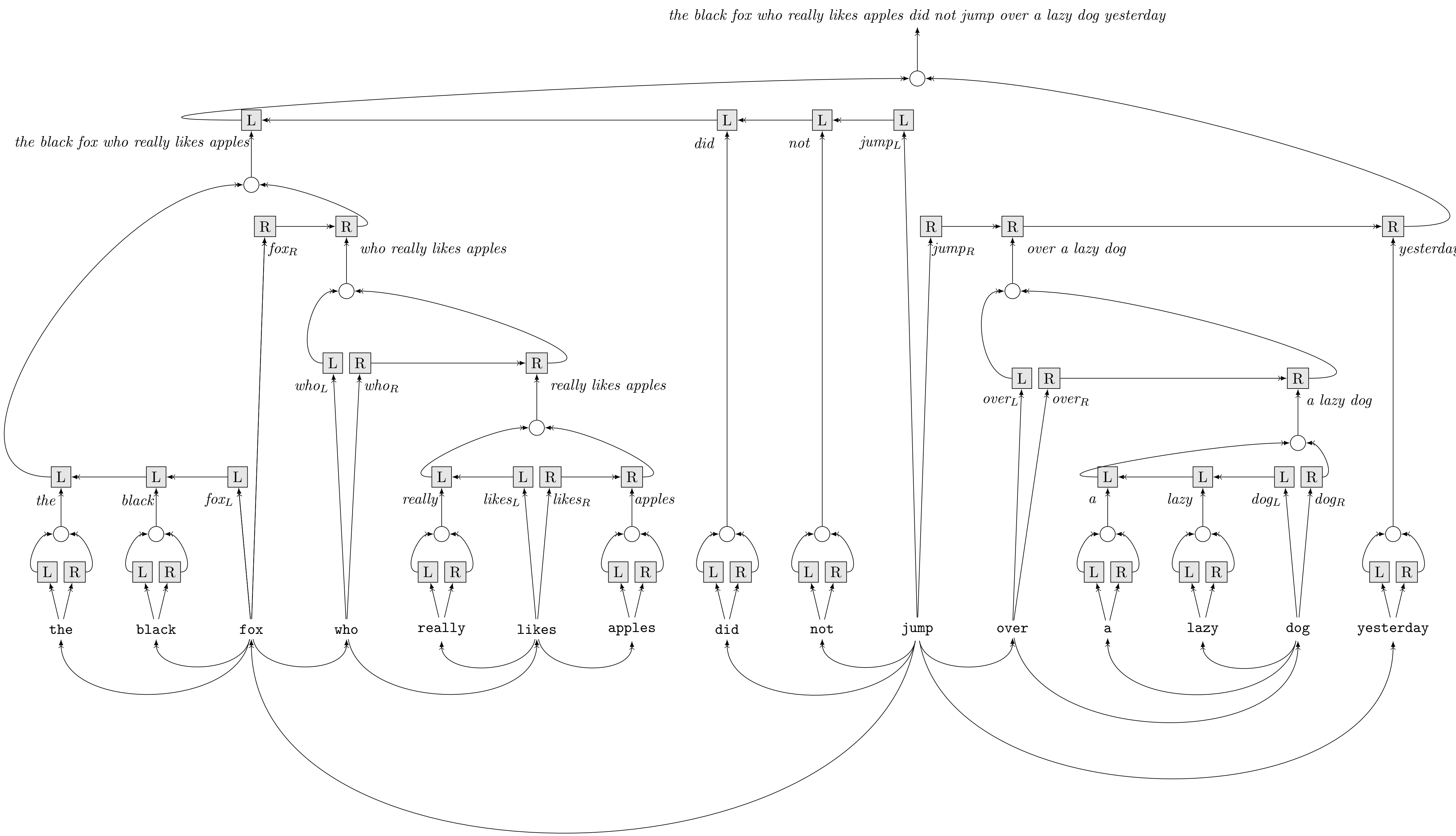


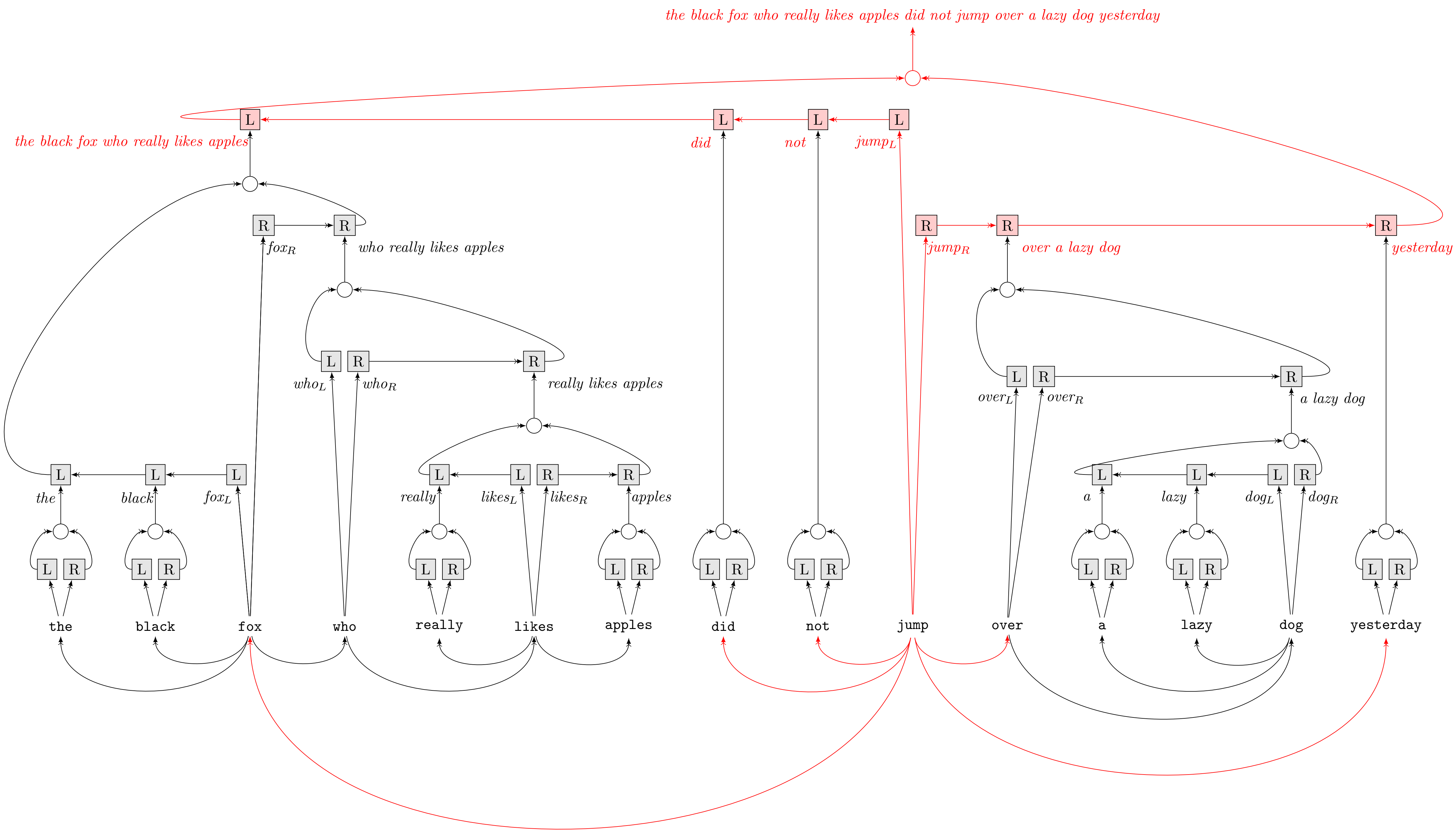
- Two LSTMs
- head + Left modifiers encoded w/ LSTM-L
- head + Right modifiers encoded w/ LSTM-R
- The Left and Right end states are concatenated

Core Idea

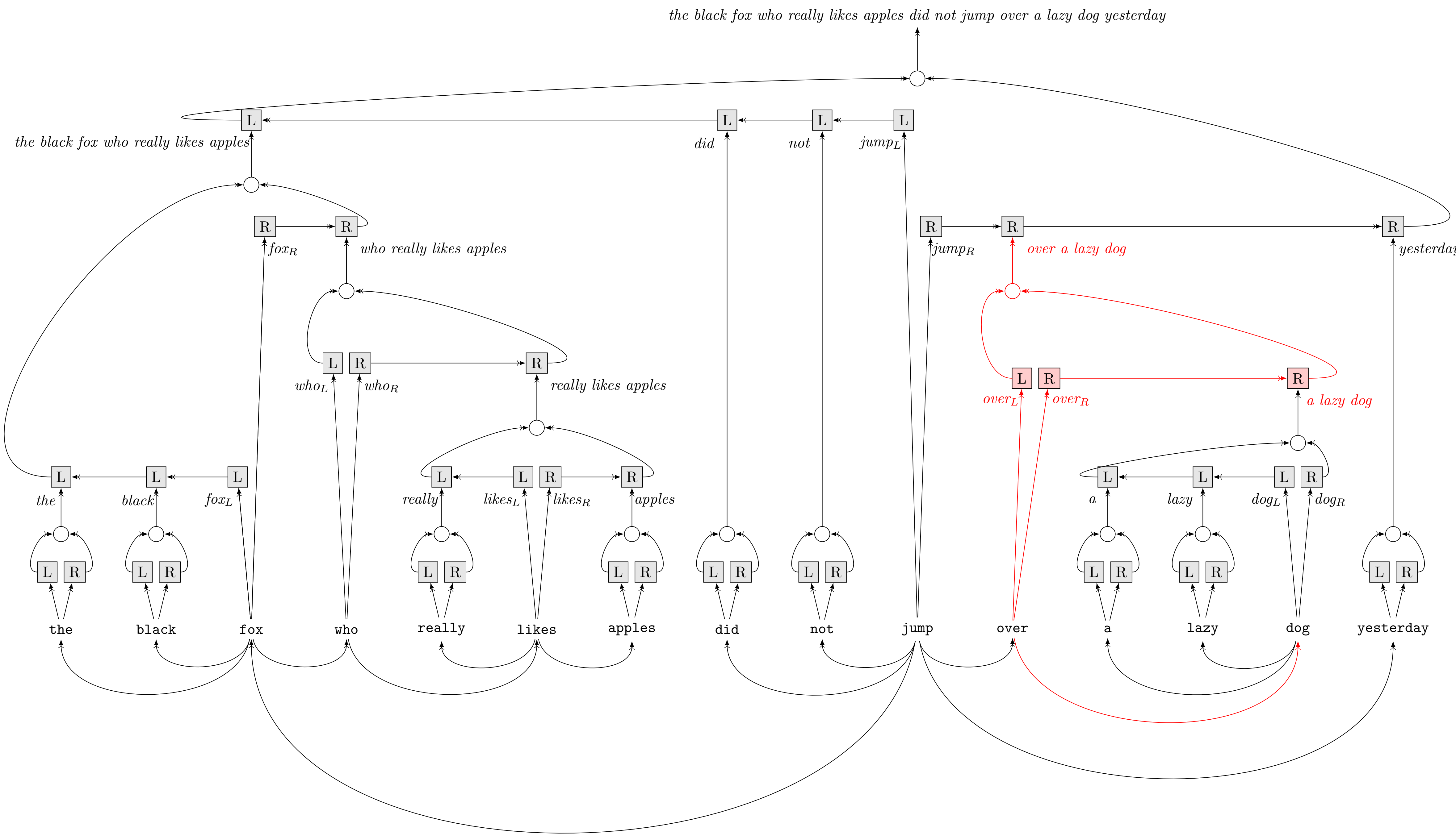


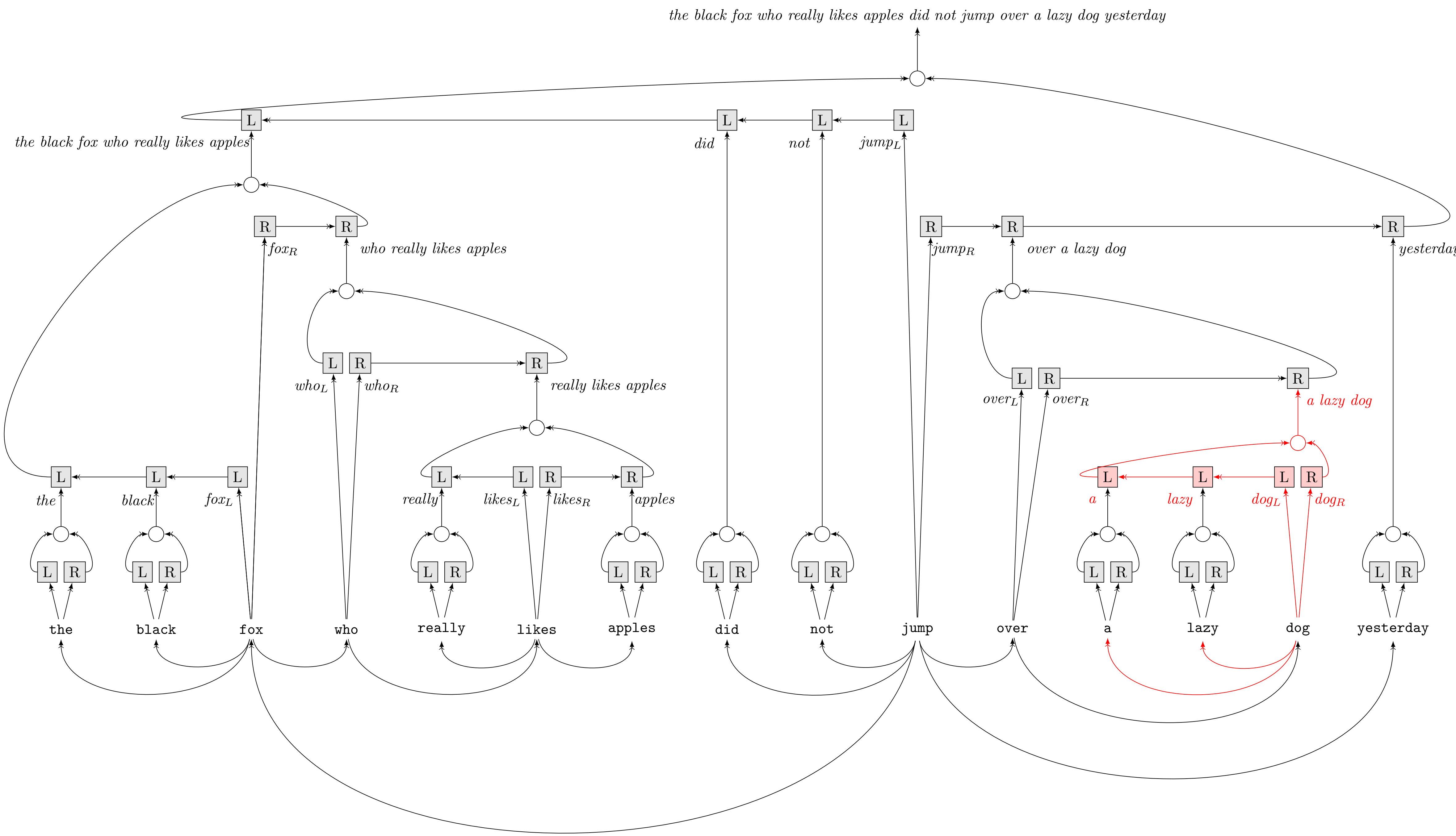
- Two LSTMs
- head + Left modifiers encoded w/ LSTM-L
- head + Right modifiers encoded w/ LSTM-R
- The Left and Right end states are concatenated

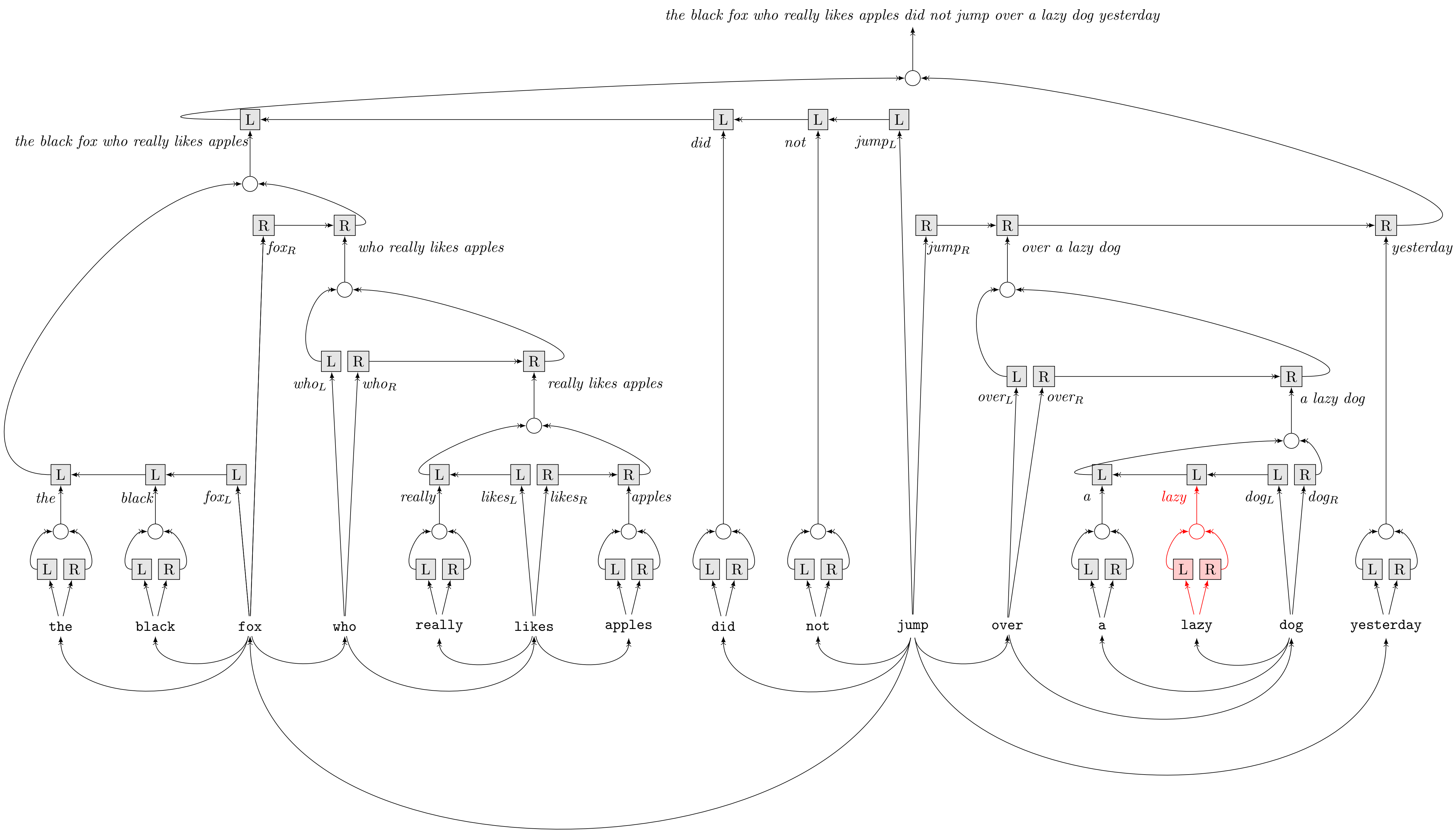

the black fox who really likes apples did not jump over a lazy dog yesterday
the black fox who really likes apples
did
not
$jump_L$
fox_R
who really likes apples
$jump_R$
over a lazy dog
yesterday
who_L
who_R
really likes apples
$over_L$
$over_R$
a lazy dog
the
black
fox_L
really
$likes_L$
$likes_R$
apples
a
lazy
dog_L
dog_R
the
black
fox
who
really
likes
apples
did
not
jump
over
a
lazy
dog
yesterday


the black fox who really likes apples did not jump over a lazy dog yesterday
the black fox who really likes apples
did
not
$jump_L$
fox_R
who really likes apples
$jump_R$
over a lazy dog
yesterday
who_L
who_R
really likes apples
$over_L$
$over_R$
a lazy dog
the
black
fox_L
really
$likes_L$
$likes_R$
apples
a
lazy
dog_L
dog_R
L
R
the
black
fox
who
really
likes
apples
did
not
jump
over
a
lazy
dog
yesterday

BIU NLP

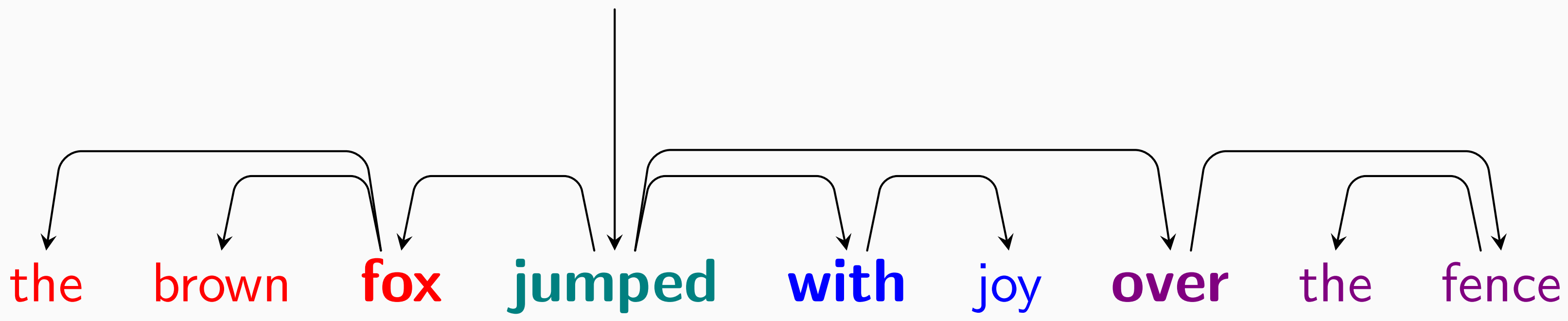

the black fox who really likes apples did not jump over a lazy dog yesterday
the black fox who really likes apples
did
not
$jump_L$
fox_R
who really likes apples
$jump_R$
over a lazy dog
yesterday
who_L
who_R
really likes apples
$over_L$
$over_R$
a lazy dog
the
black
fox_L
really
$likes_L$
$likes_R$
apples
a
lazy
dog_L
dog_R
L
R
the
black
fox
who
really
likes
apples
did
not
jump
over
a
lazy
dog
yesterday


the black fox who really likes apples did not jump over a lazy dog yesterday
the black fox who really likes apples
did
not
jump_L
fox_R
who really likes apples
jump_R
over a lazy dog
yesterday
who_L
who_R
really likes apples
over_L
over_R
a lazy dog
the
black
fox_L
really
likes_L
likes_R
apples
a
lazy
dog_L
dog_R
L
R
the
black
fox
who
really
likes
apples
did
not
jump
over
a
lazy
dog
yesterday

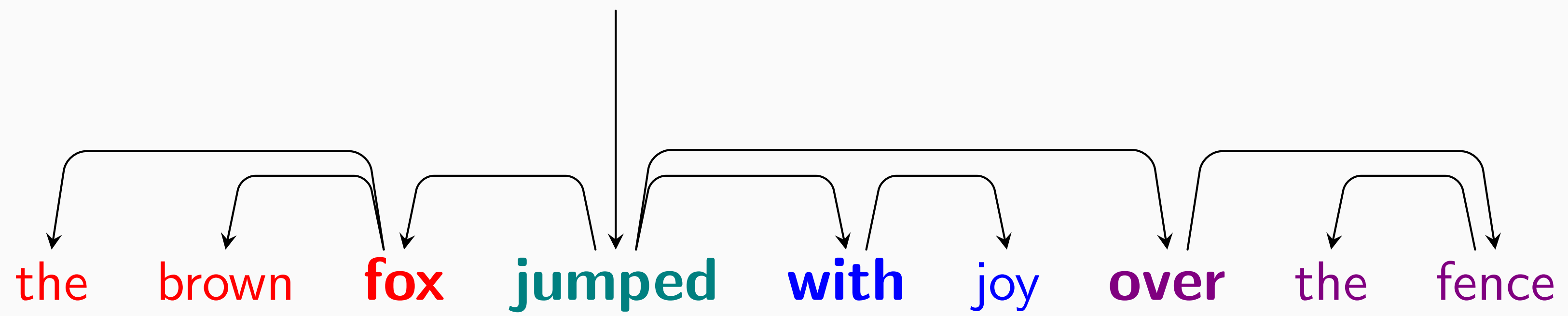


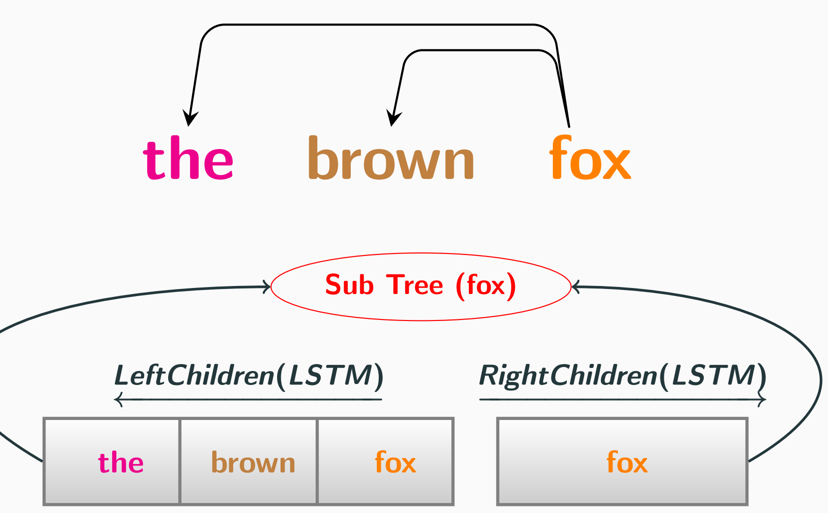
the black fox who really likes apples did not jump over a lazy dog yesterday
the black fox who really likes apples
did
not
$jump_L$
fox_R
who really likes apples
$jump_R$
over a lazy dog
yesterday
who_L
who_R
really likes apples
$over_L$
$over_R$
a lazy dog
the
black
fox_L
really
$likes_L$
$likes_R$
apples
a
lazy
dog_L
dog_R
the
black
fox
who
really
likes
apples
did
not
jump
over
a
lazy
dog
yesterday

Hierarchical Tree LSTM (Example)

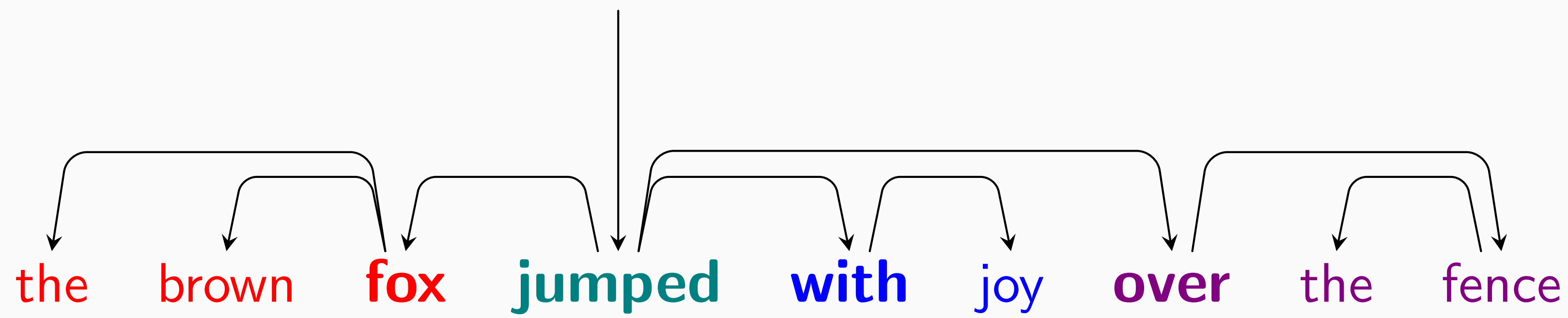


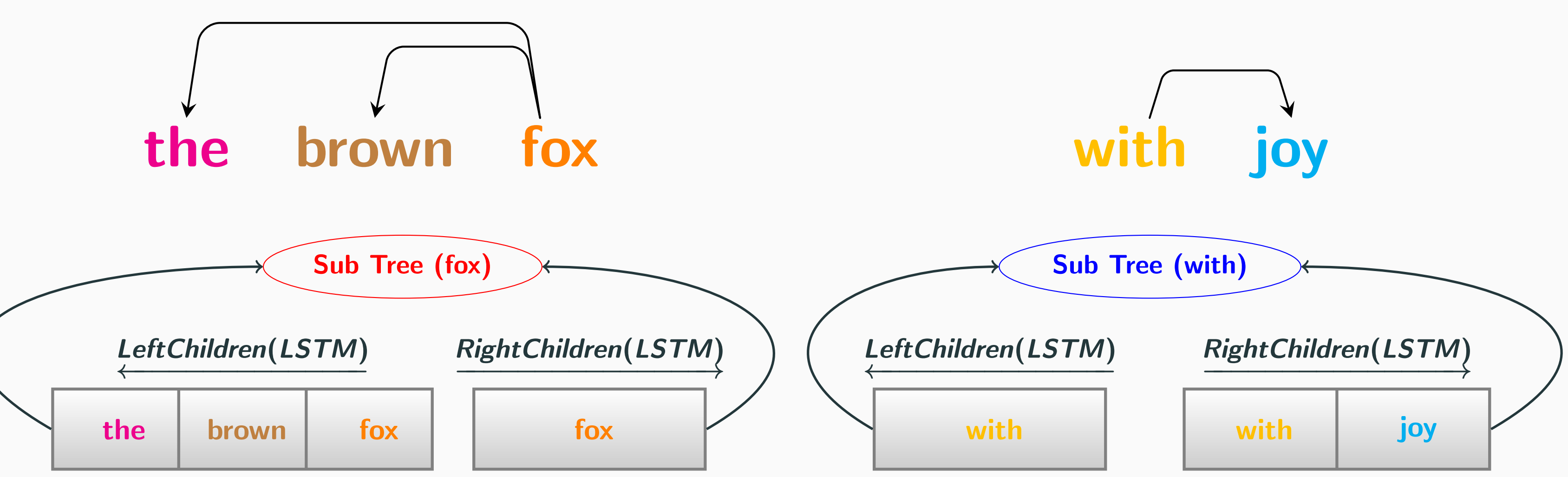
Hierarchical Tree LSTM (Example)



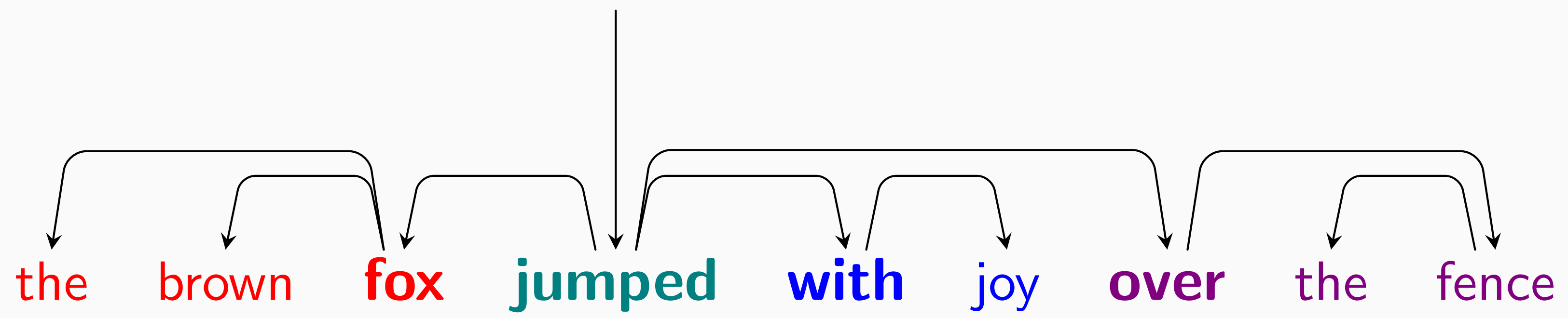


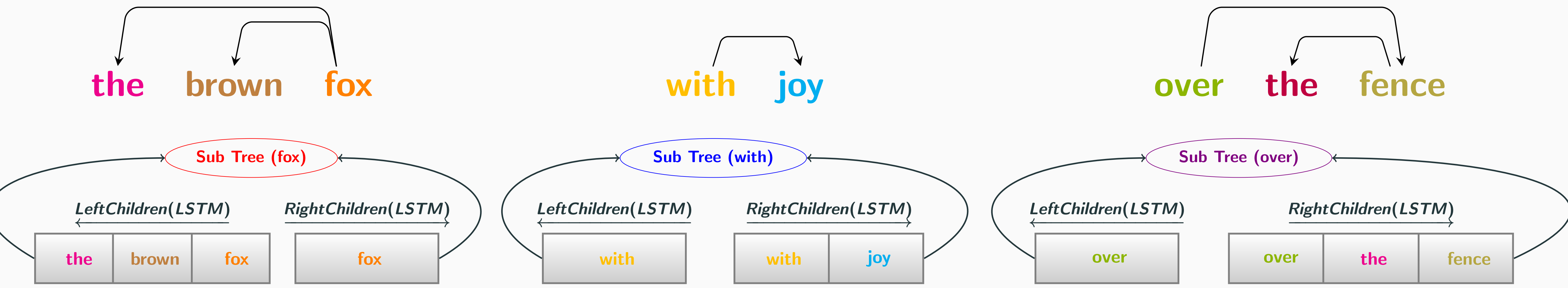
Hierarchical Tree LSTM (Example)



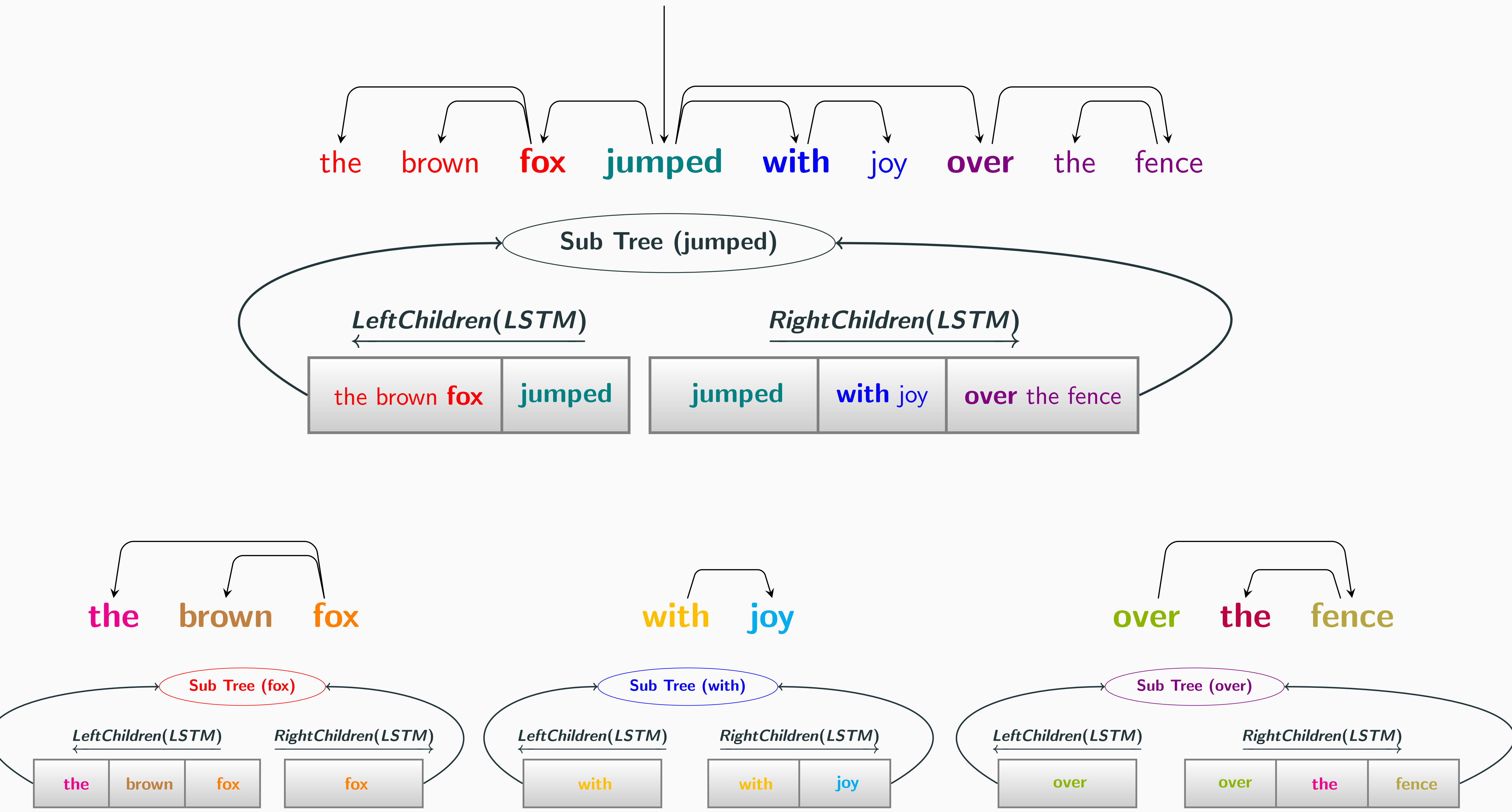


Hierarchical Tree LSTM (Example)

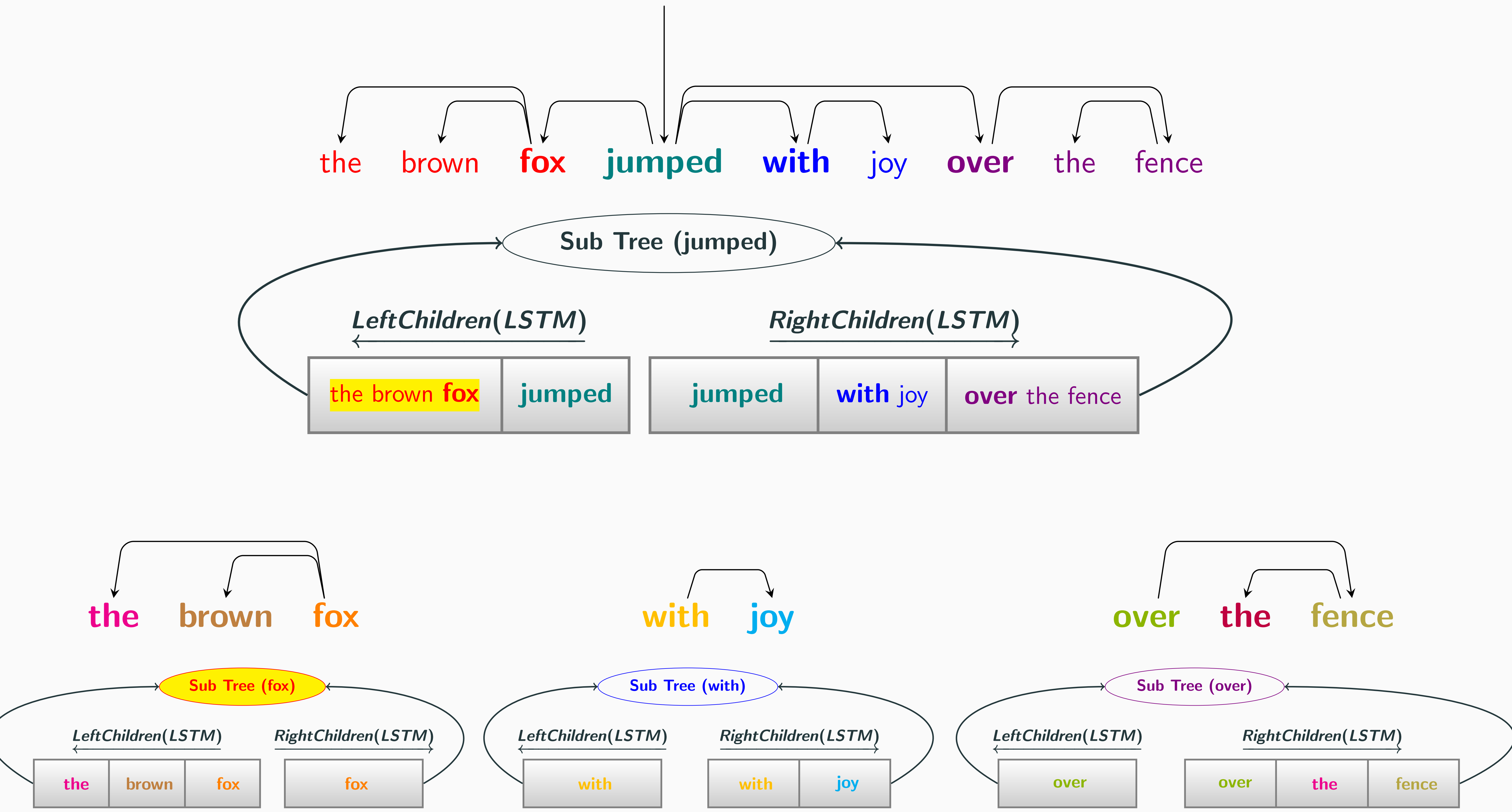




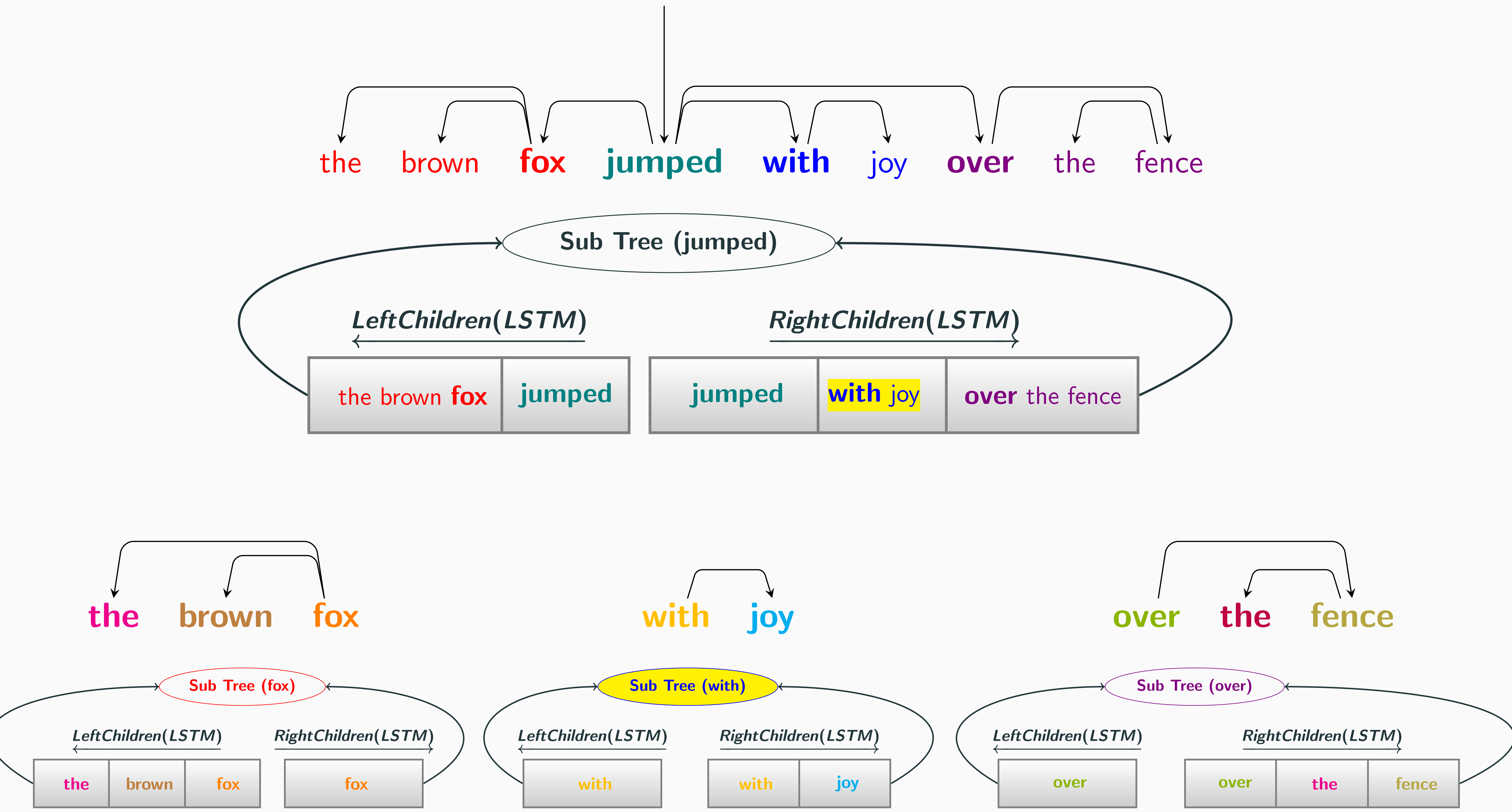
Hierarchical Tree LSTM (Example)



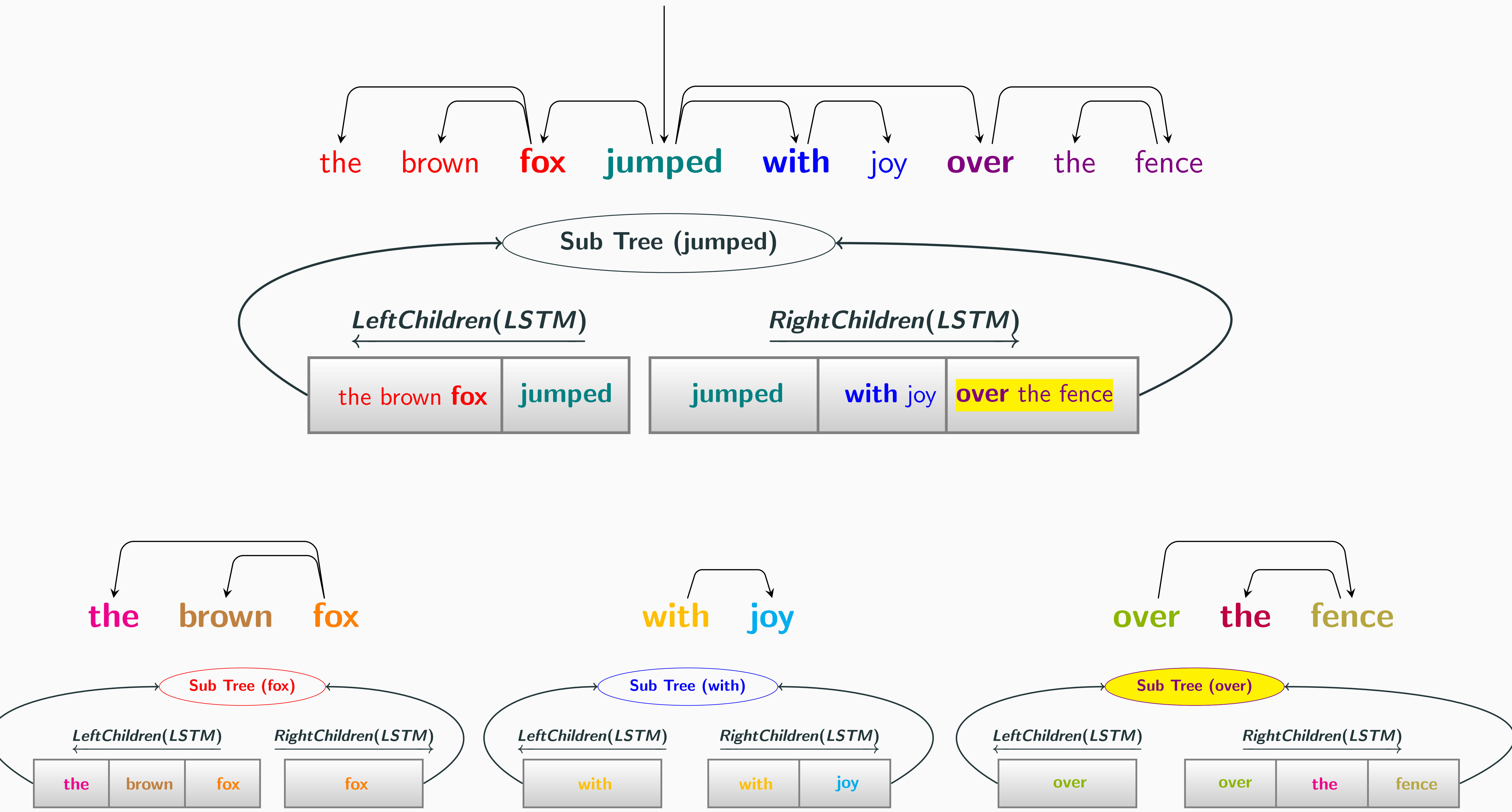
Hierarchical Tree LSTM (Example)



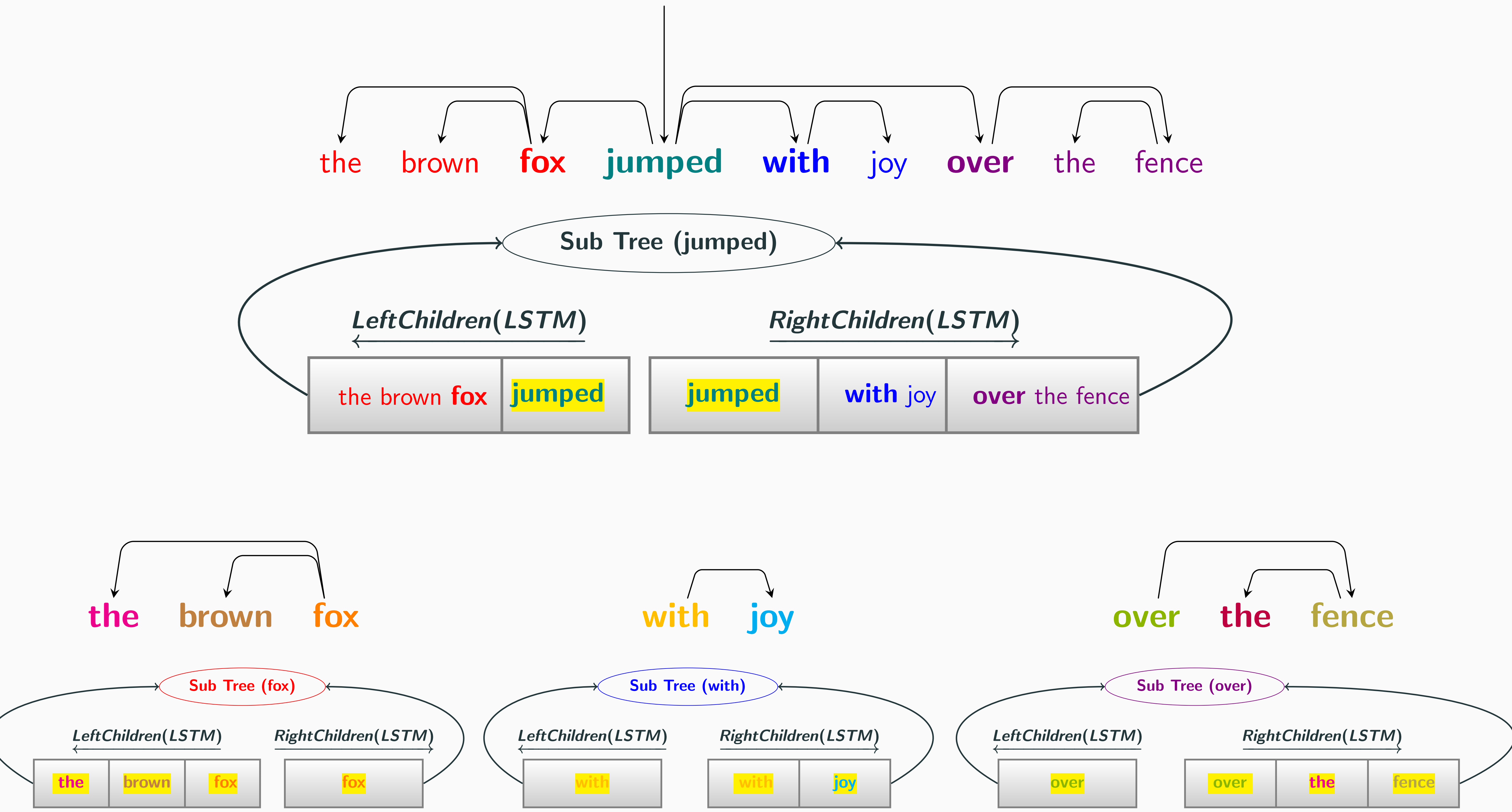
Hierarchical Tree LSTM (Example)



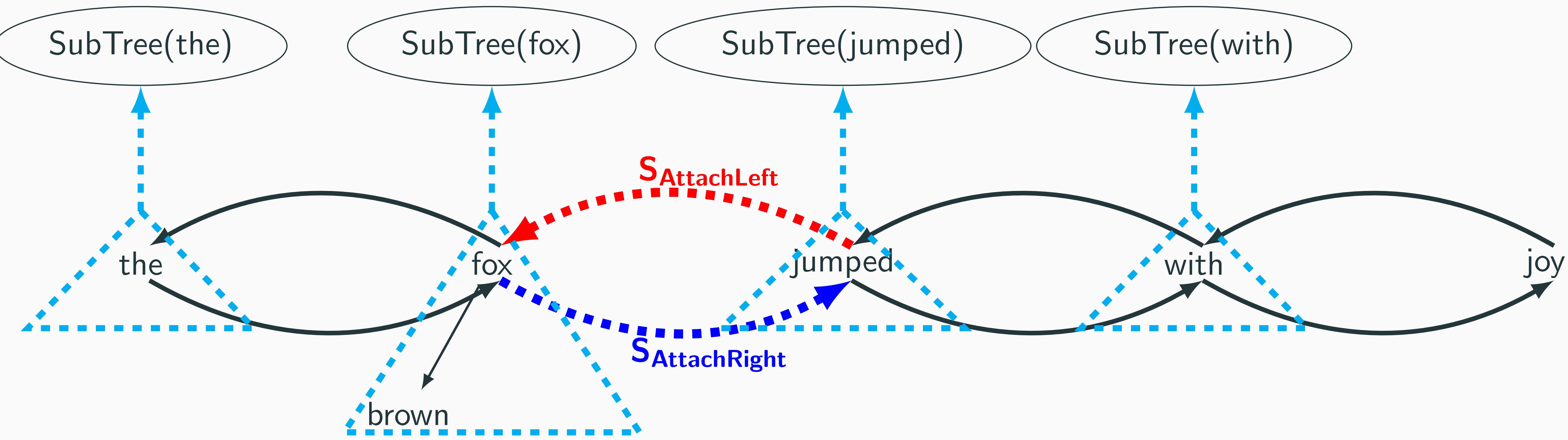
Hierarchical Tree LSTM (Example)



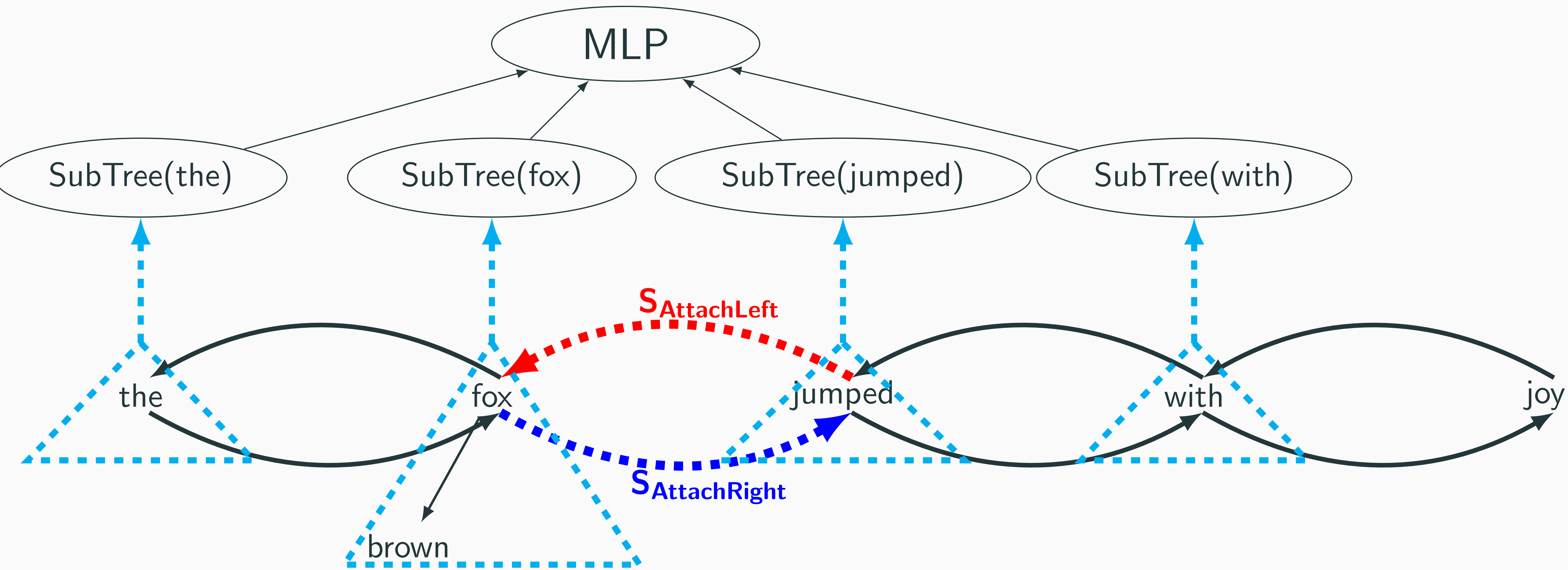
How Do We Capture Leafs?



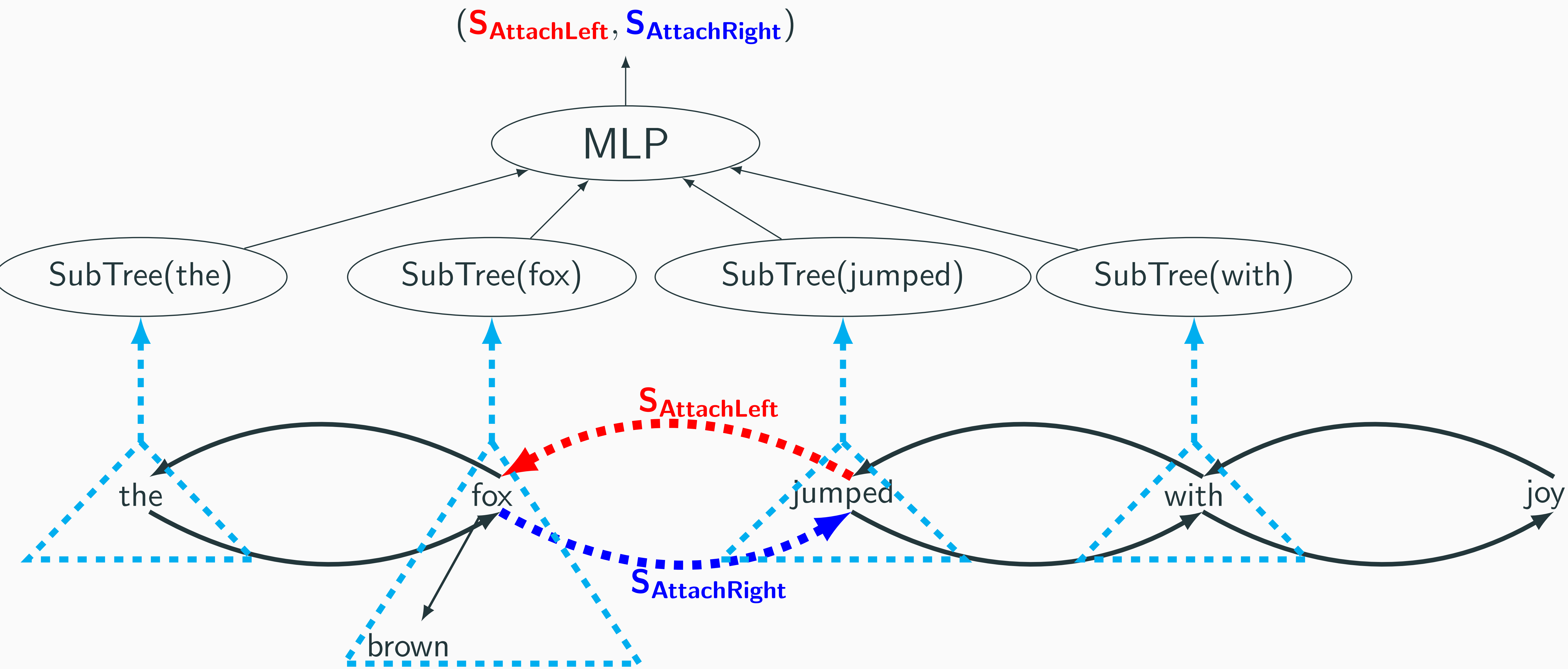
Easy-First Parsing (Score Function)



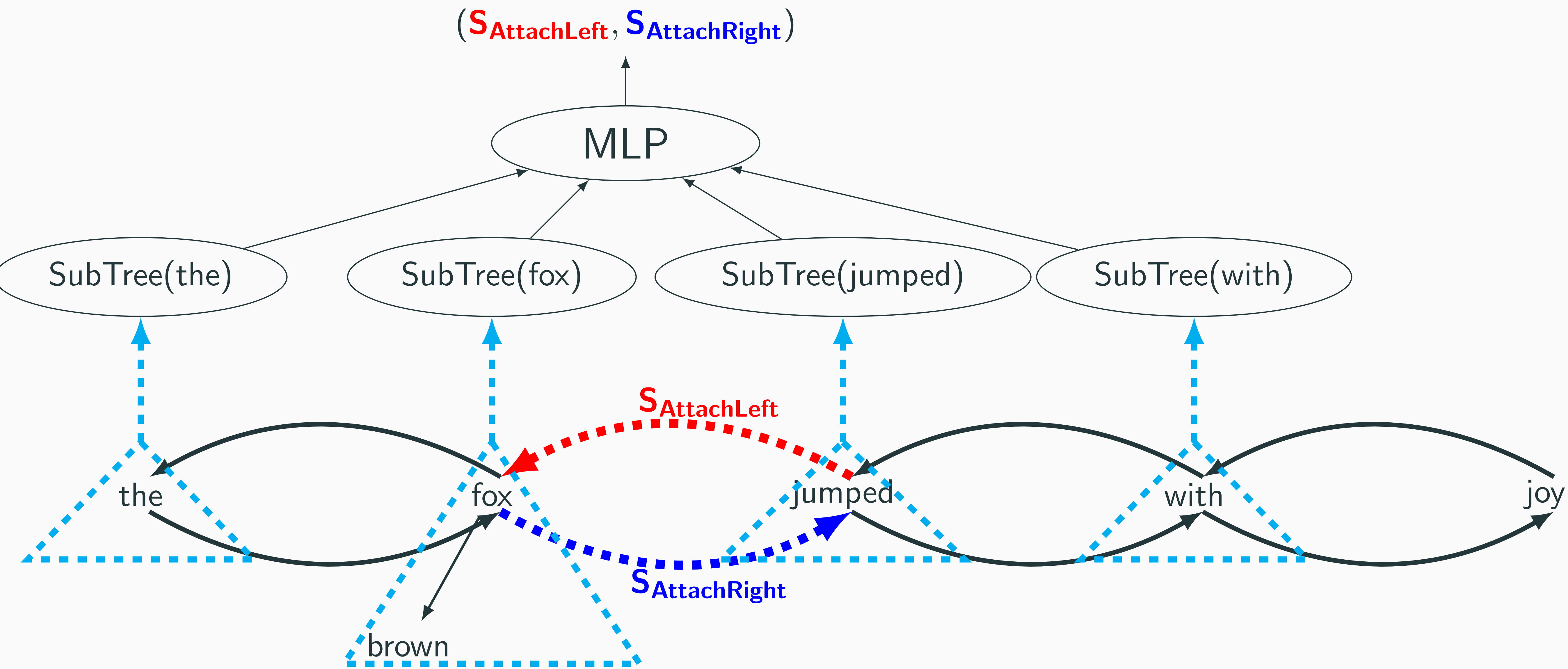
Easy-First Parsing (Score Function)



Easy-First Parsing (Score Function)



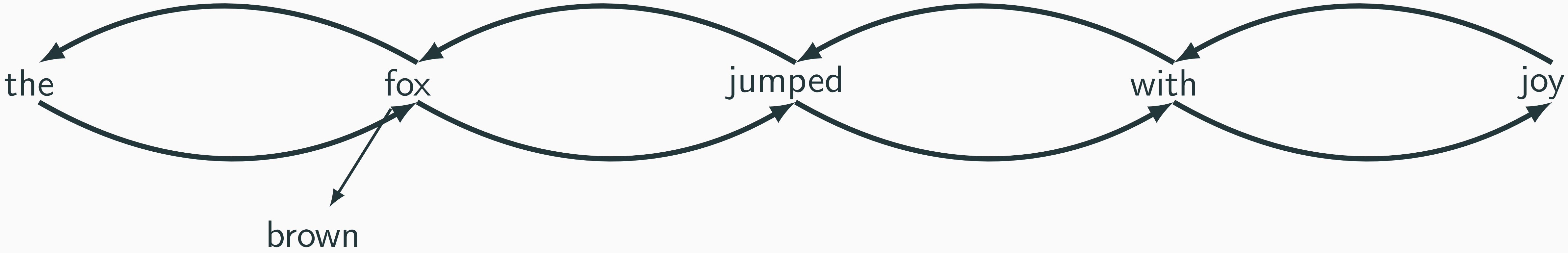
Easy-First Parsing (Score Function)



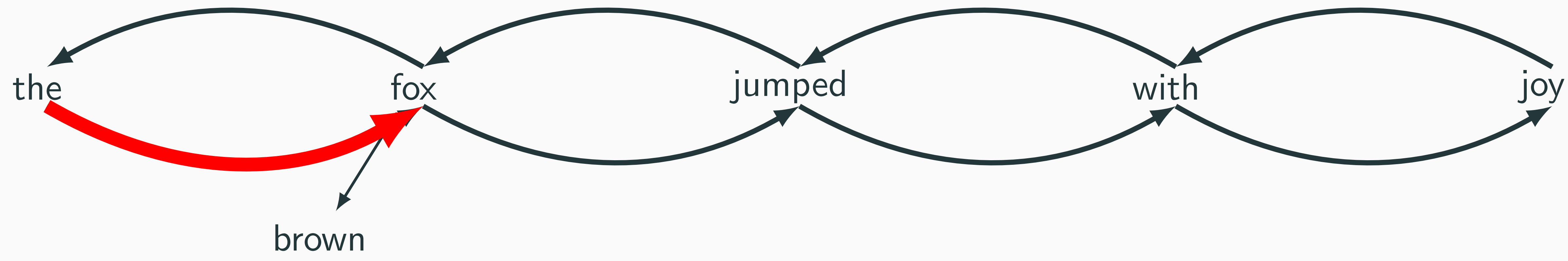
How do we efficiently compute the Hierarchical Tree LSTM representation for each pending sub-tree?

Bottom-Up Tree Representation Building

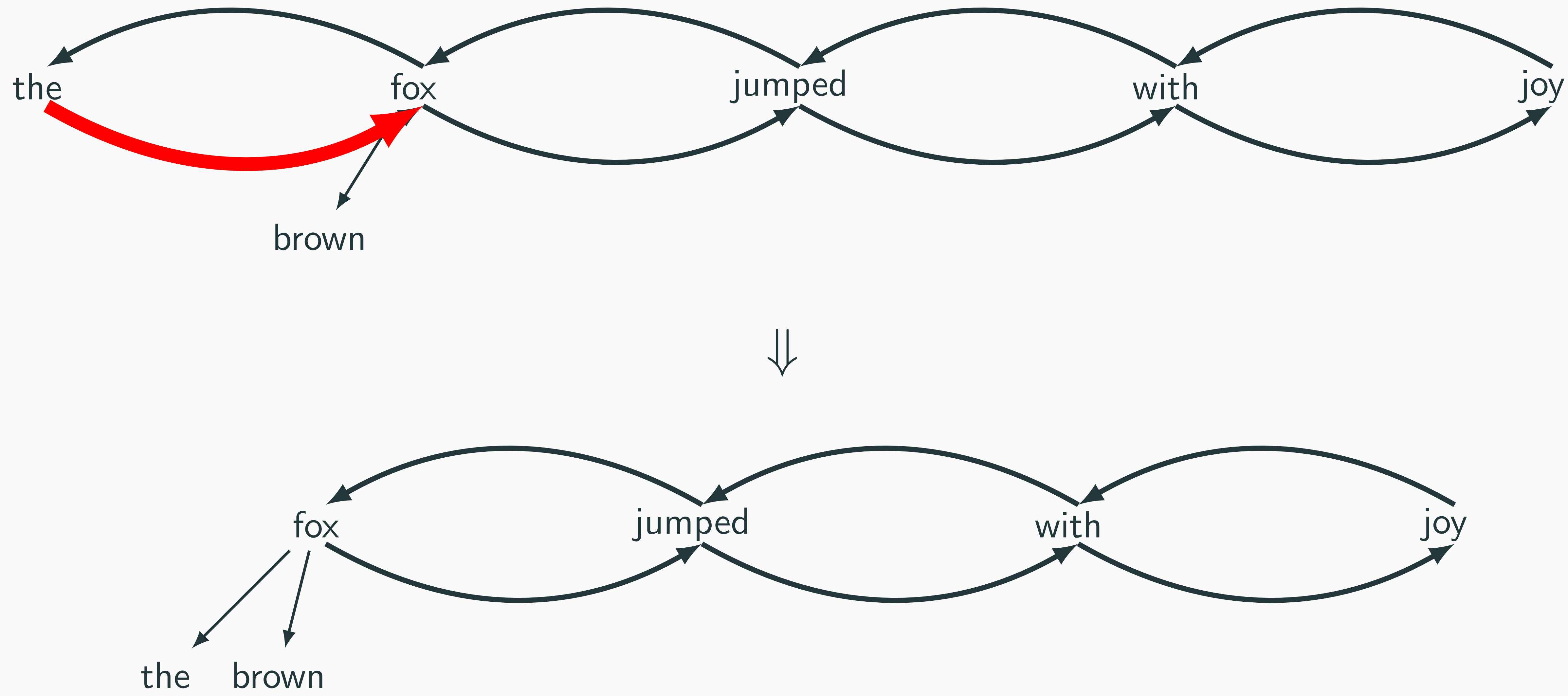
Bottom-Up Tree Representation Building



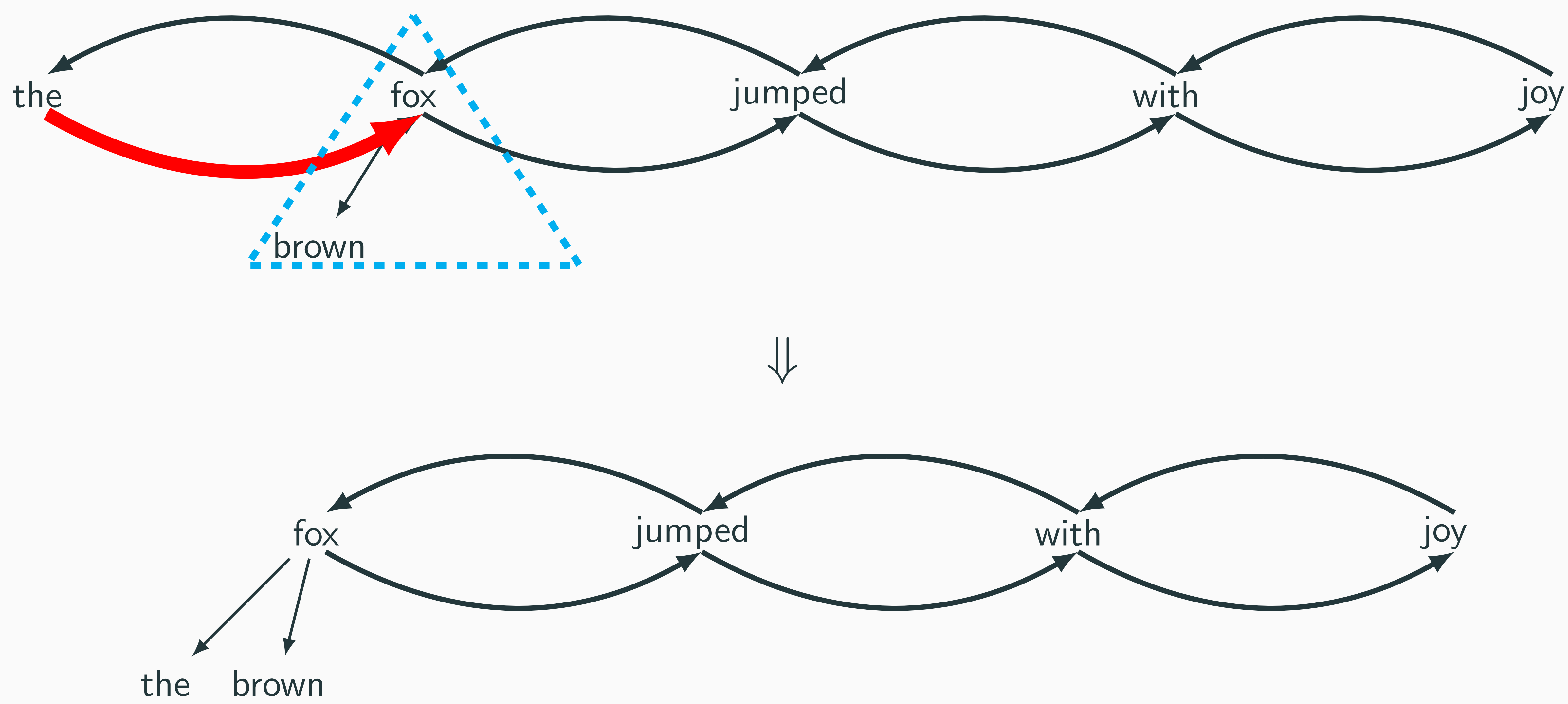
Bottom-Up Tree Representation Building

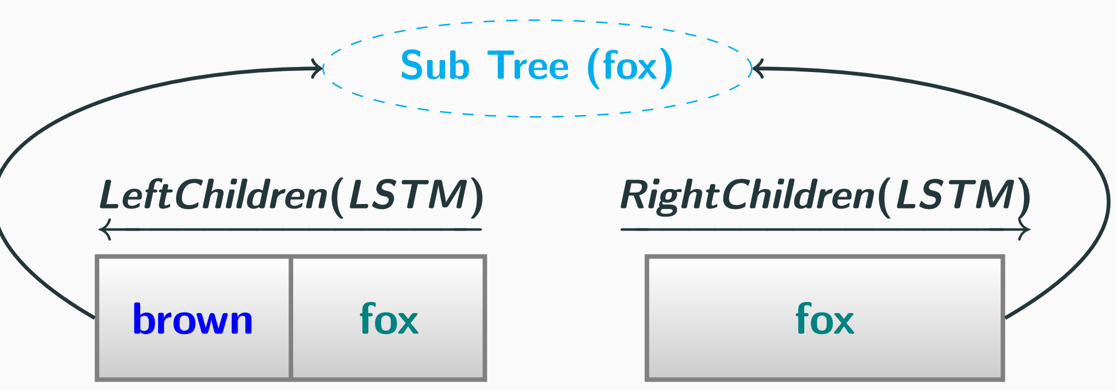


Bottom-Up Tree Representation Building

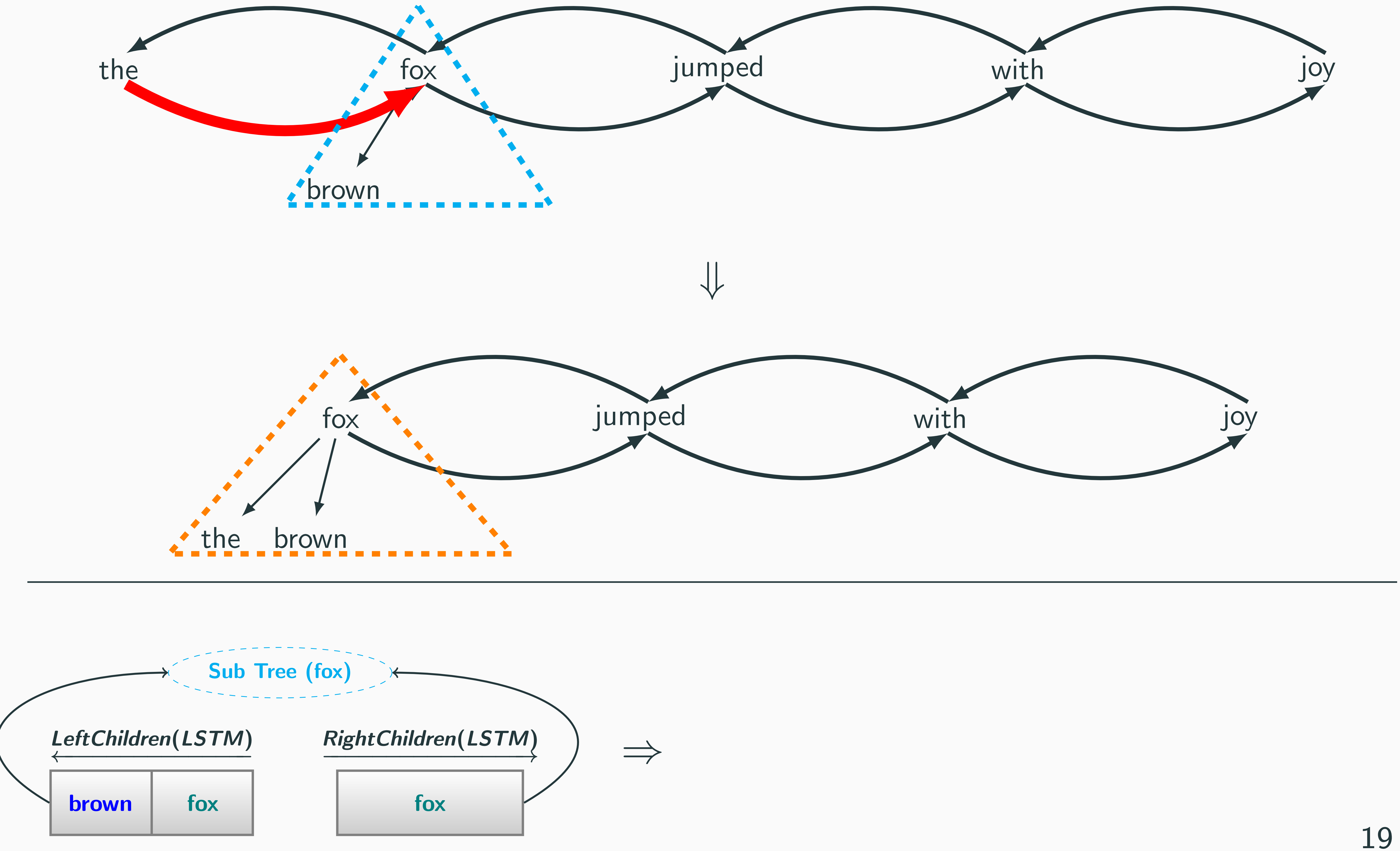


Bottom-Up Tree Representation Building

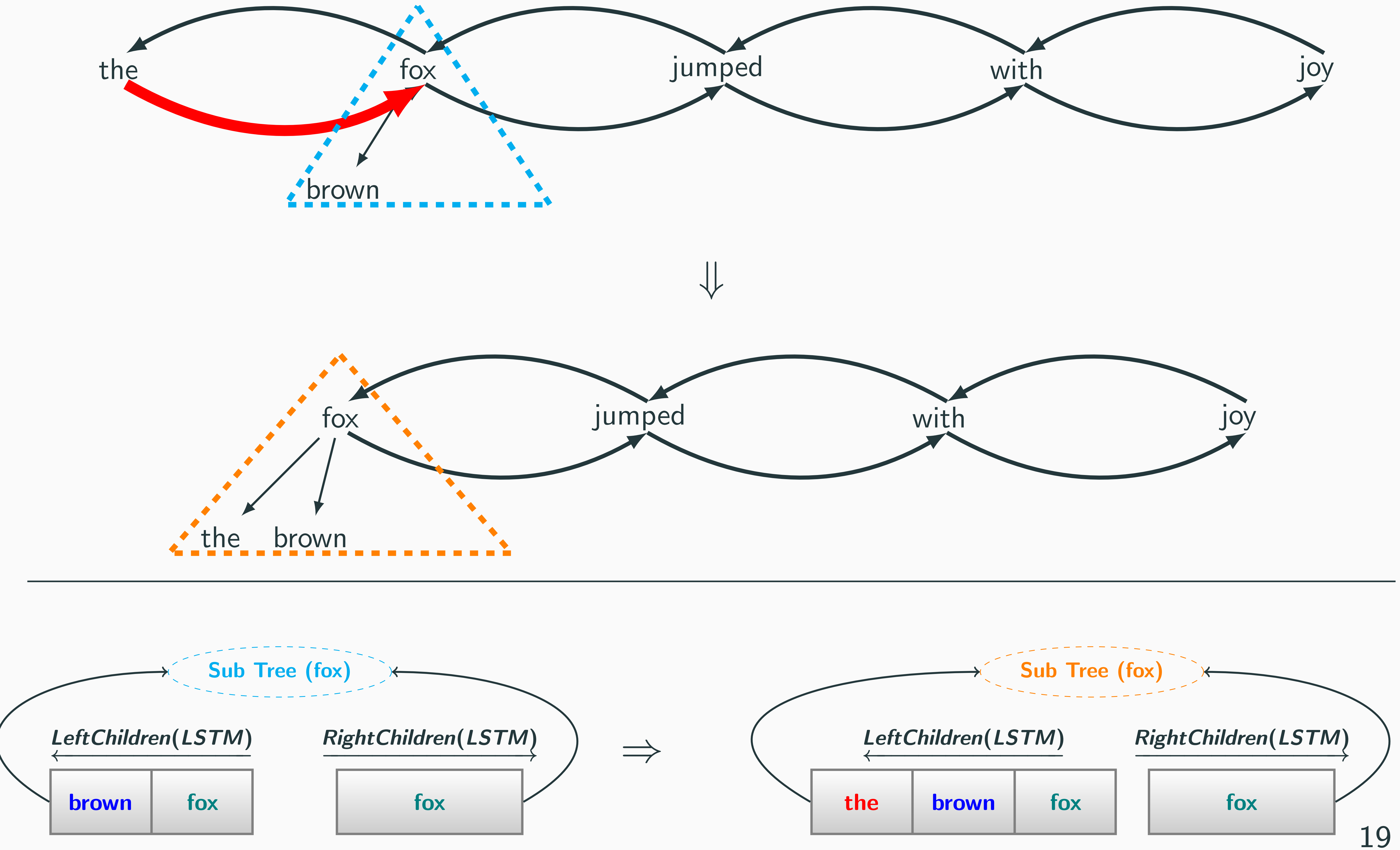




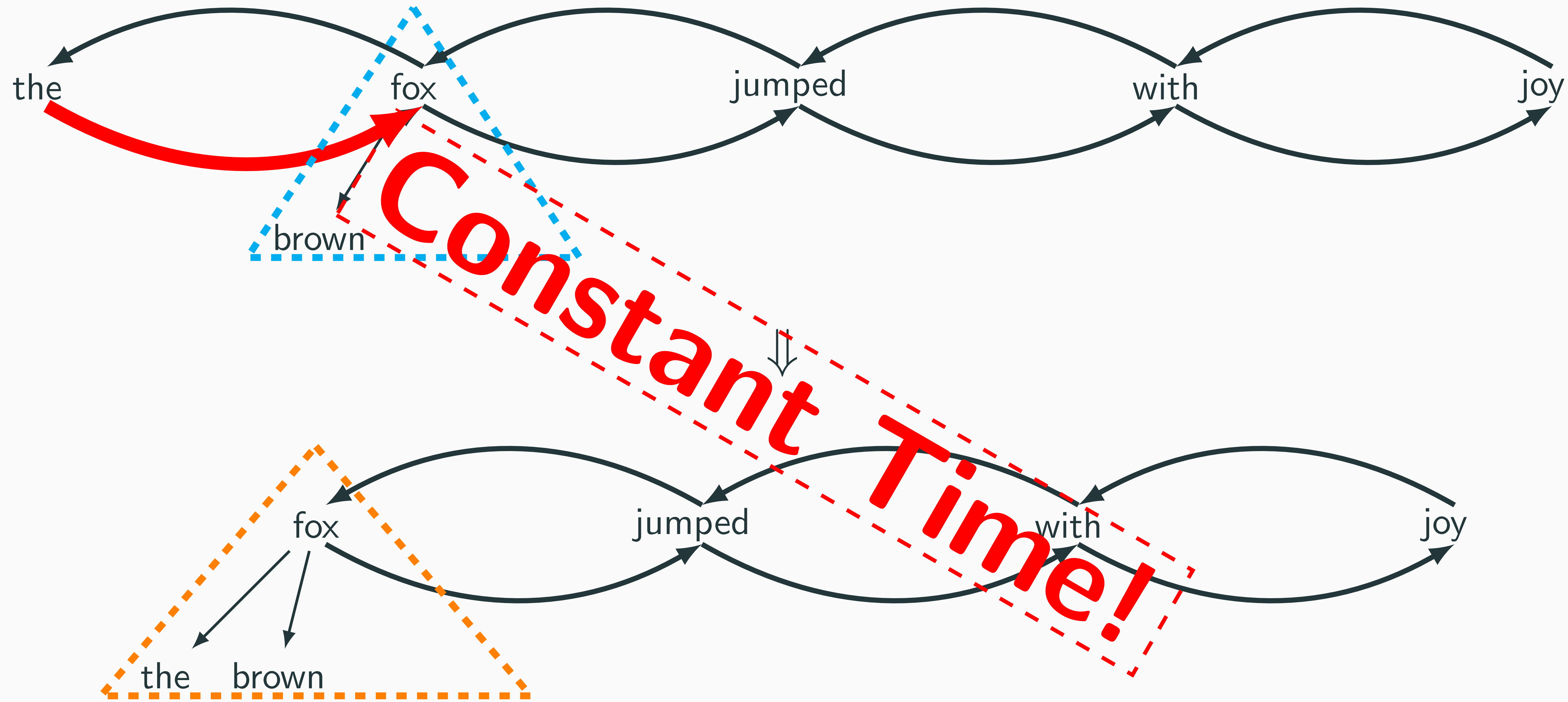
Bottom-Up Tree Representation Building

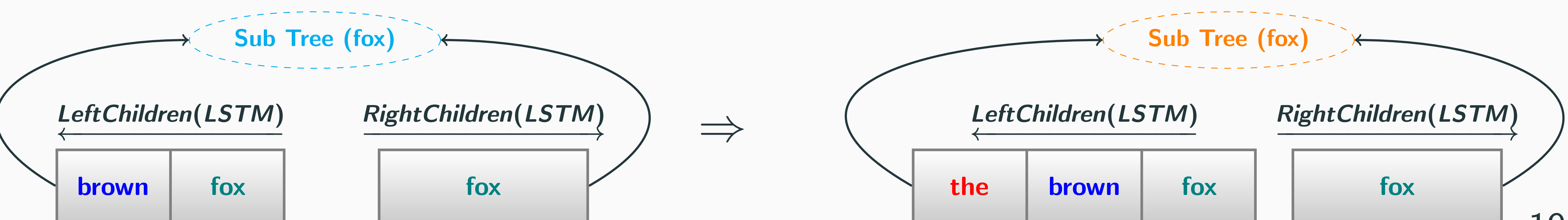


Bottom-Up Tree Representation Building

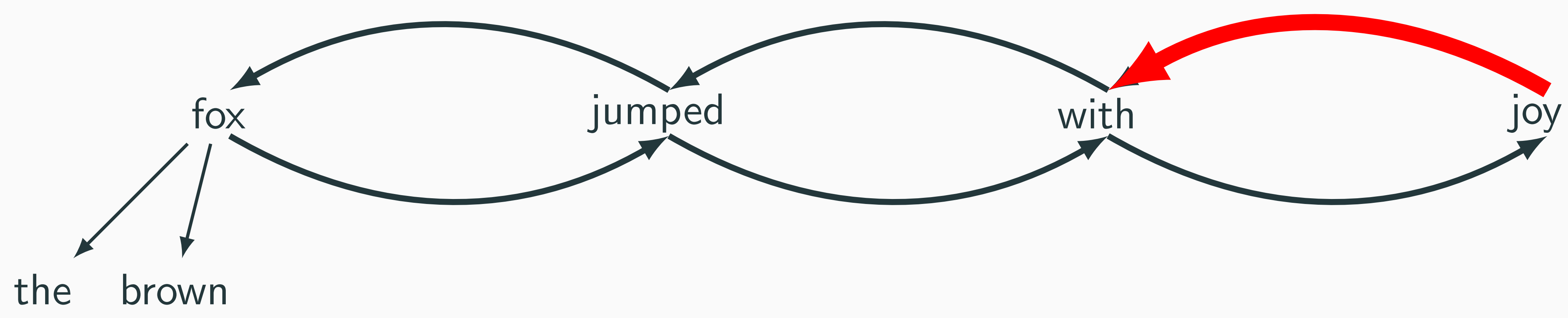


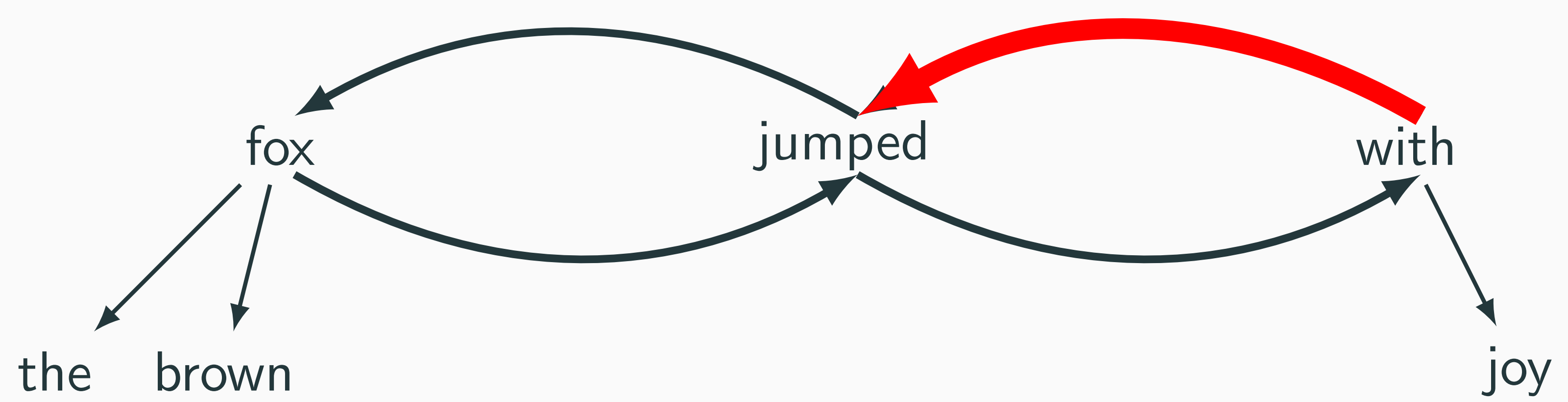
Bottom-Up Tree Representation Building

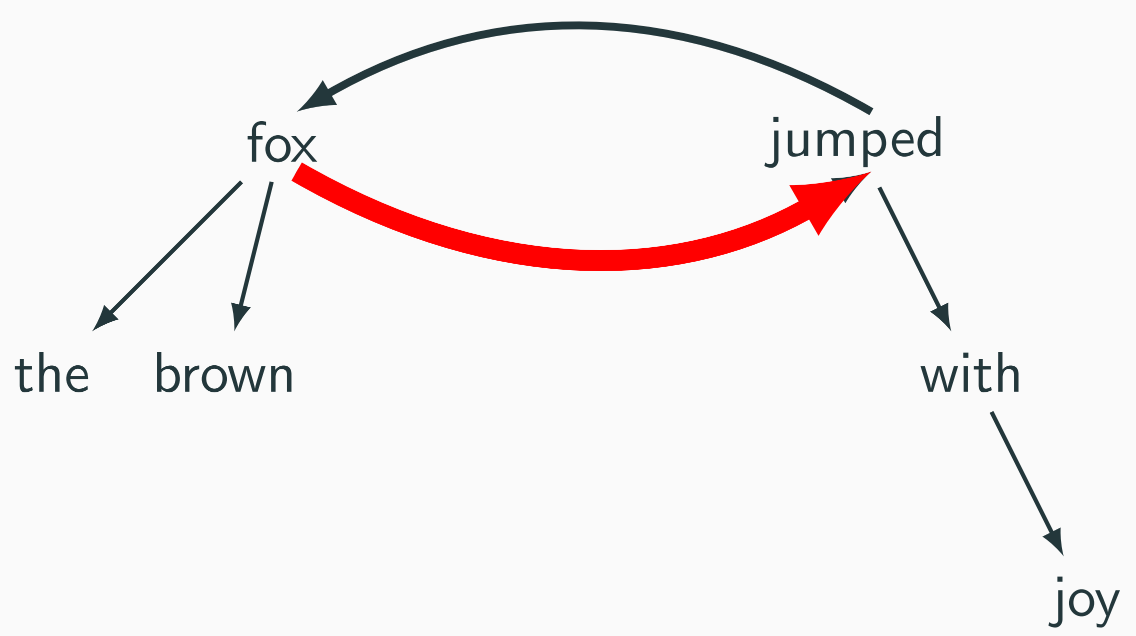


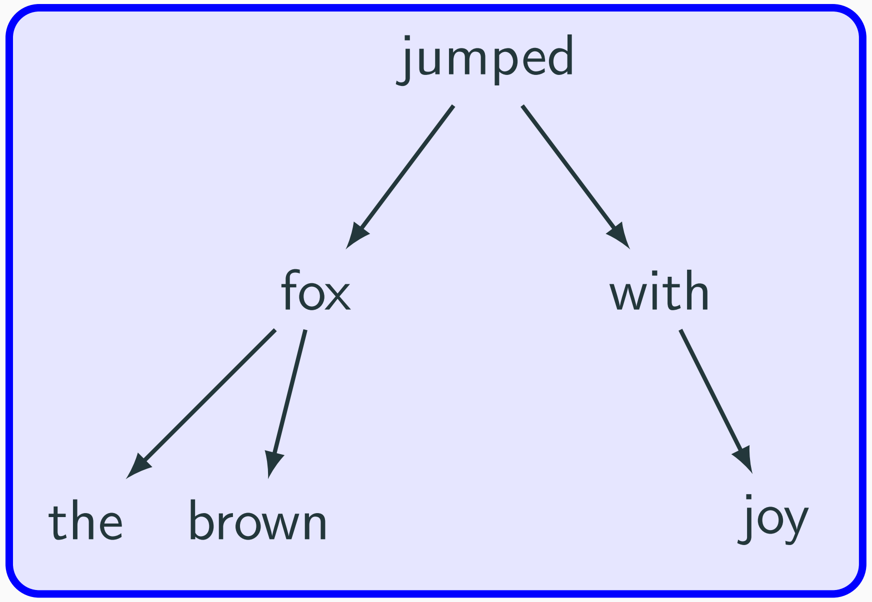


Easy-First Parsing (Continued)









Easy First Parsing with Hierarchical Tree LSTMs

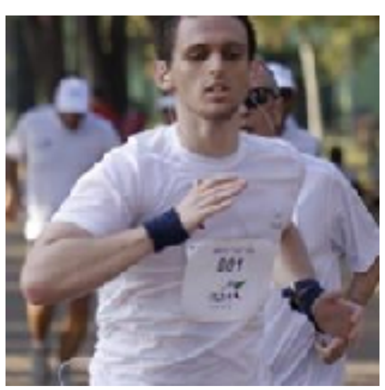
- It works! We get nice results.
- **But**.
- Turns out can actually do something much simpler.

Parsing with LSTMs, Take 2

Simple and Accurate Dependency Parsing
Using Bidirectional LSTM Feature Representations

Eliyahu Kiperwasser
Computer Science Department
Bar-Ilan University
Ramat-Gan, Israel
elikip@gmail.com

Yoav Goldberg
Computer Science Department
Bar-Ilan University
Ramat-Gan, Israel
yoav.goldberg@gmail.com

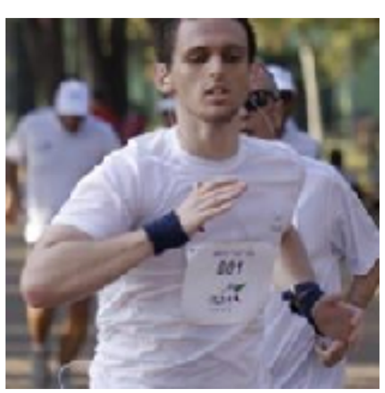


Parsing with LSTMs, Take 2

Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations

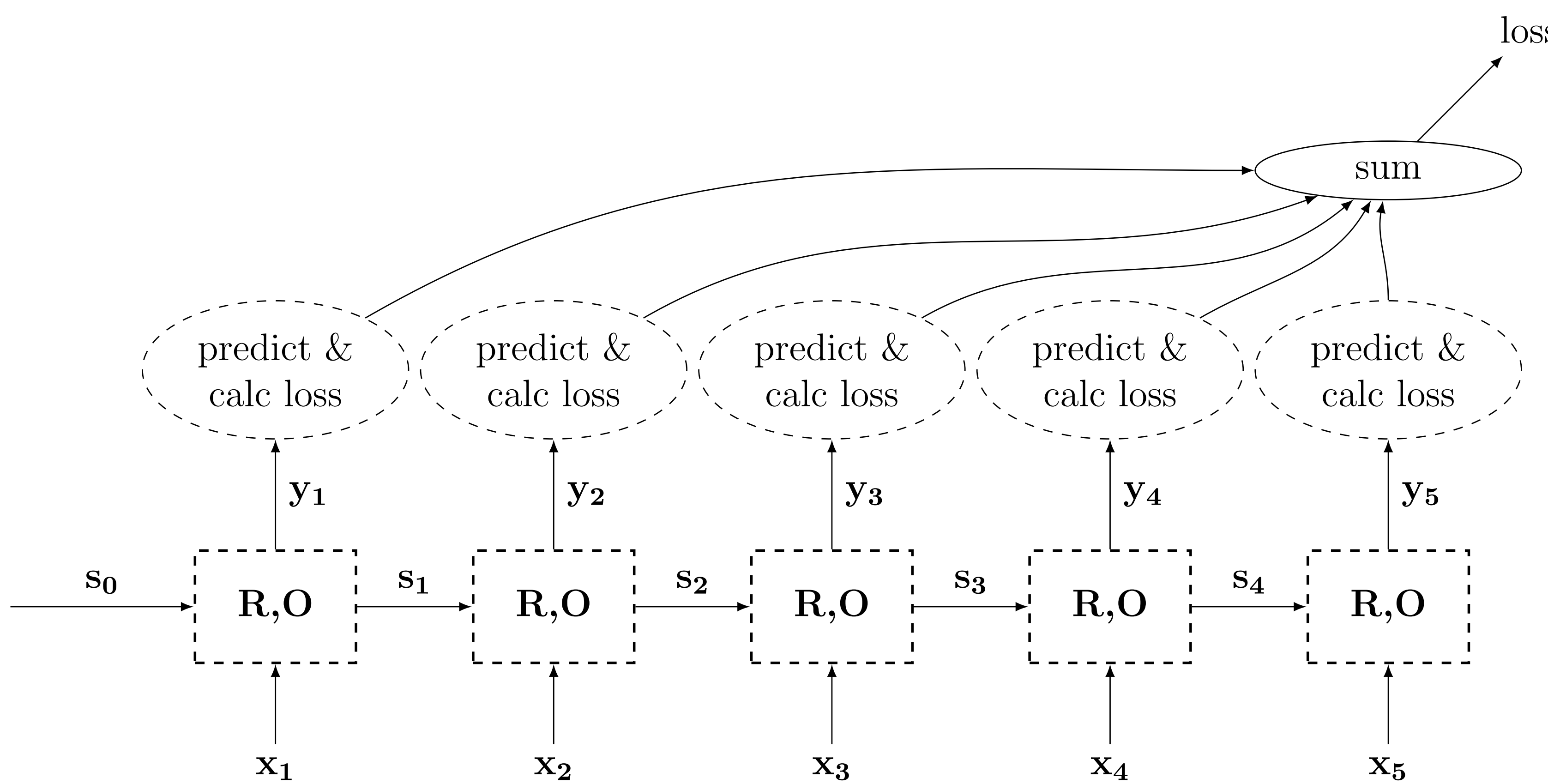
Eliyahu Kiperwasser
Computer Science Department
Bar-Ilan University
Ramat-Gan, Israel
elikip@gmail.com

Yoav Goldberg
Computer Science Department
Bar-Ilan University
Ramat-Gan, Israel
yoav.goldberg@gmail.com



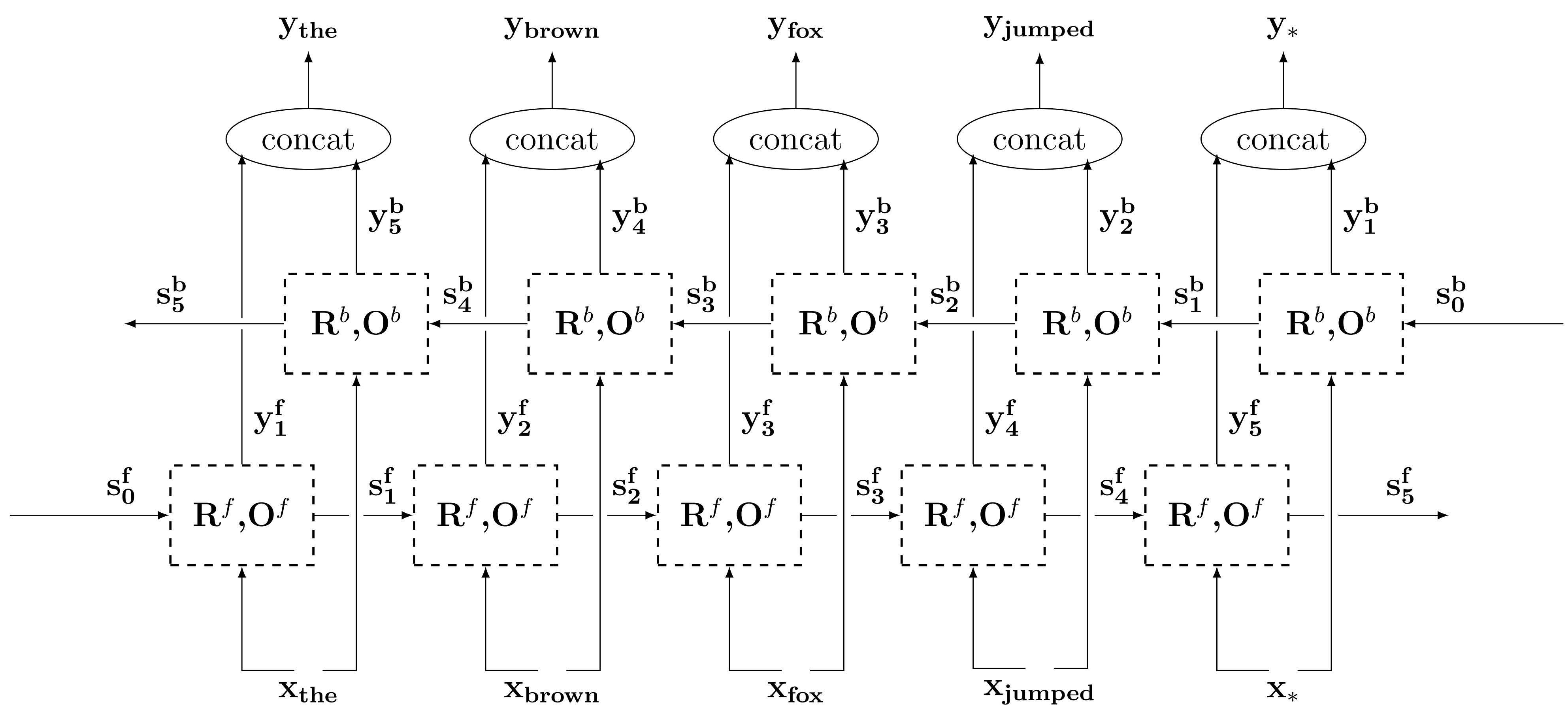
Bi-directional RNNs

RNNs so far:



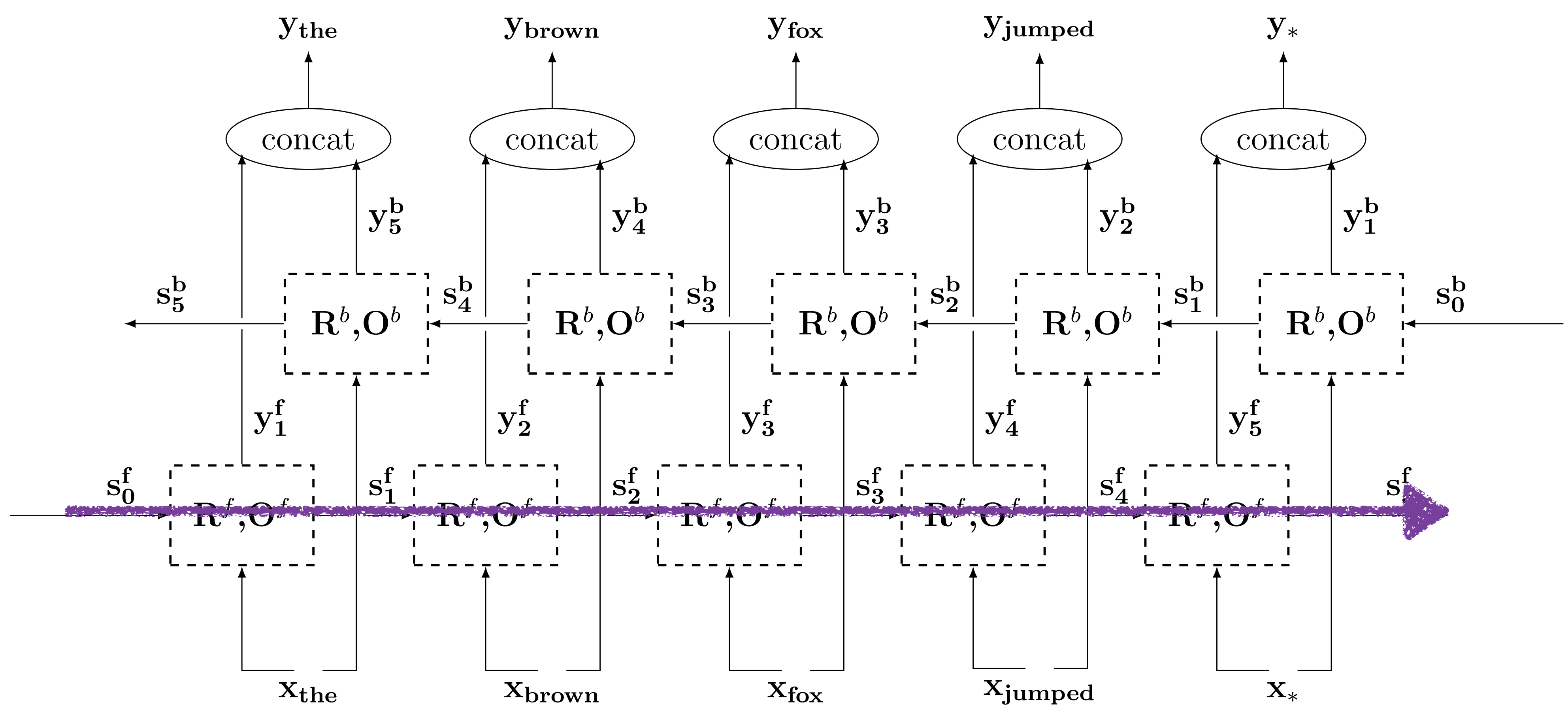
Each state encodes the entire history up to that state.
This is not bad. **But what about the future?**

Bidirectional RNNs



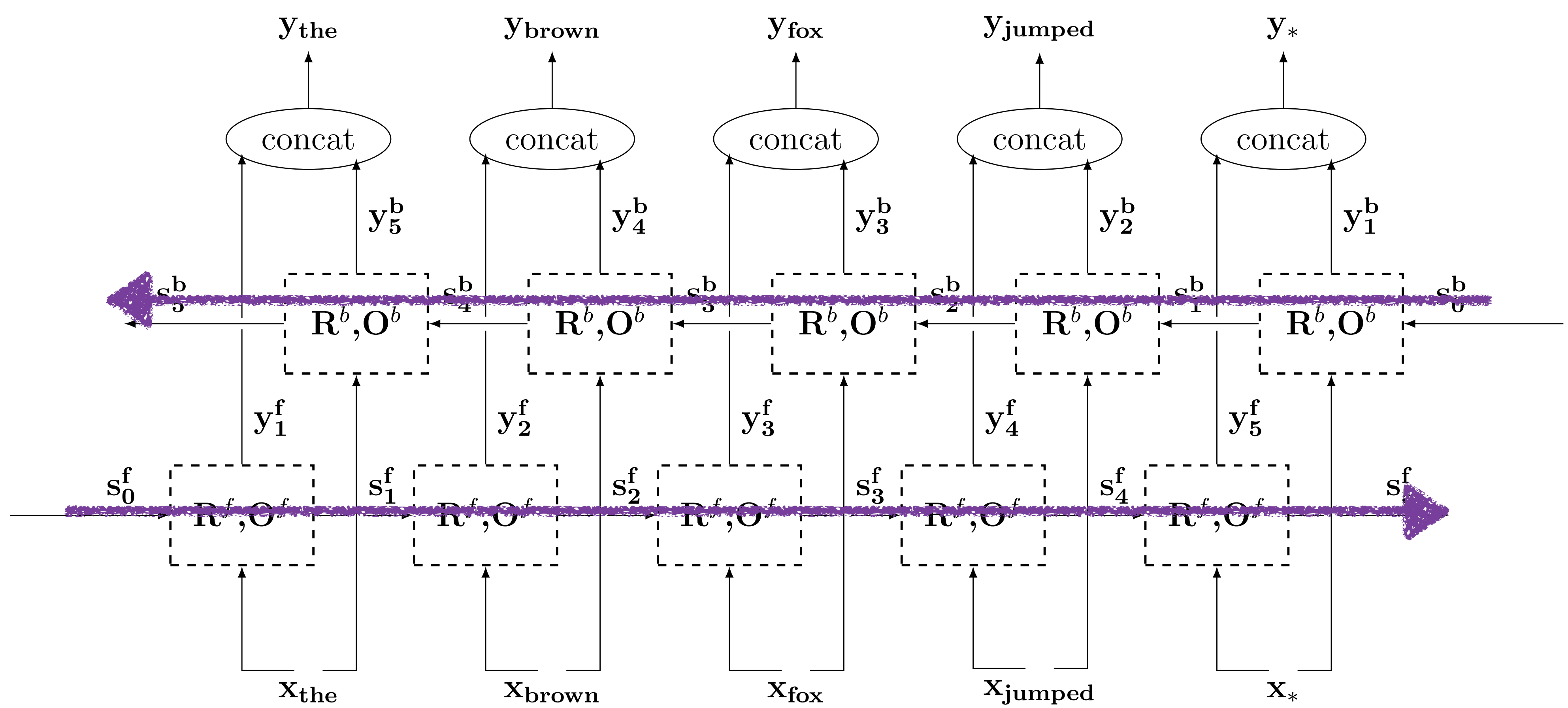
One RNN runs left to right.
Another runs right to left.
Encode **both future and history** of a word.

Bidirectional RNNs



One RNN runs left to right.
Another runs right to left.
Encode **both future and history** of a word.

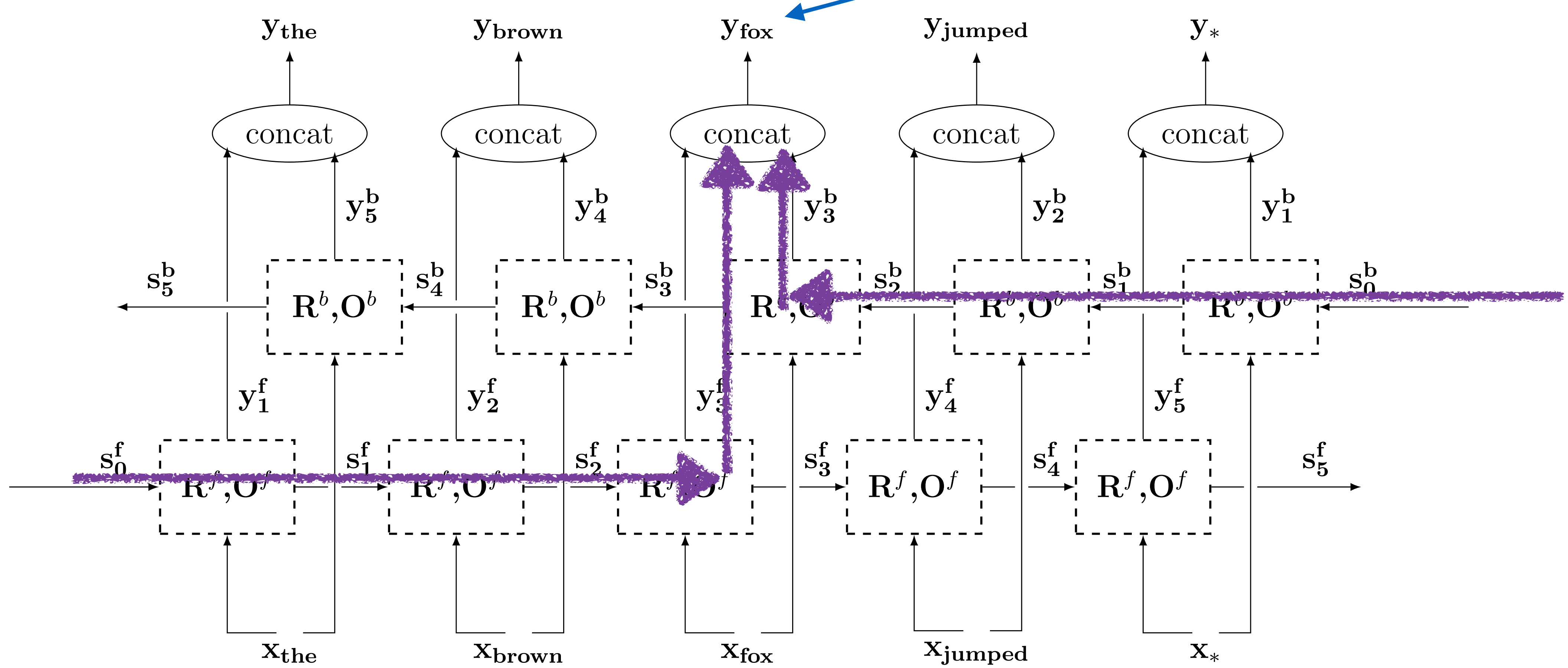
Bidirectional RNNs



One RNN runs left to right.
Another runs right to left.
Encode **both future and history** of a word.

BI-RNNs

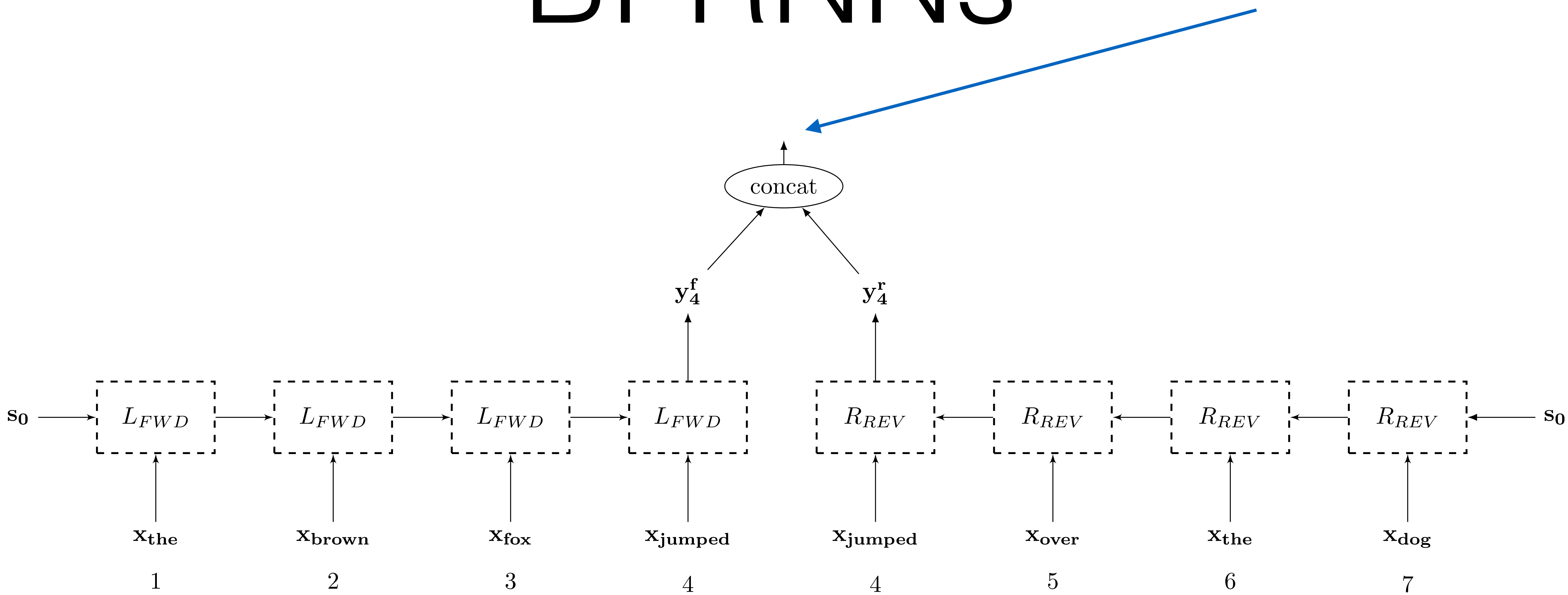
an infinite window around the word.



One RNN runs left to right.
Another runs right to left.
Encode **both future and history** of a word.

BI-RNNs

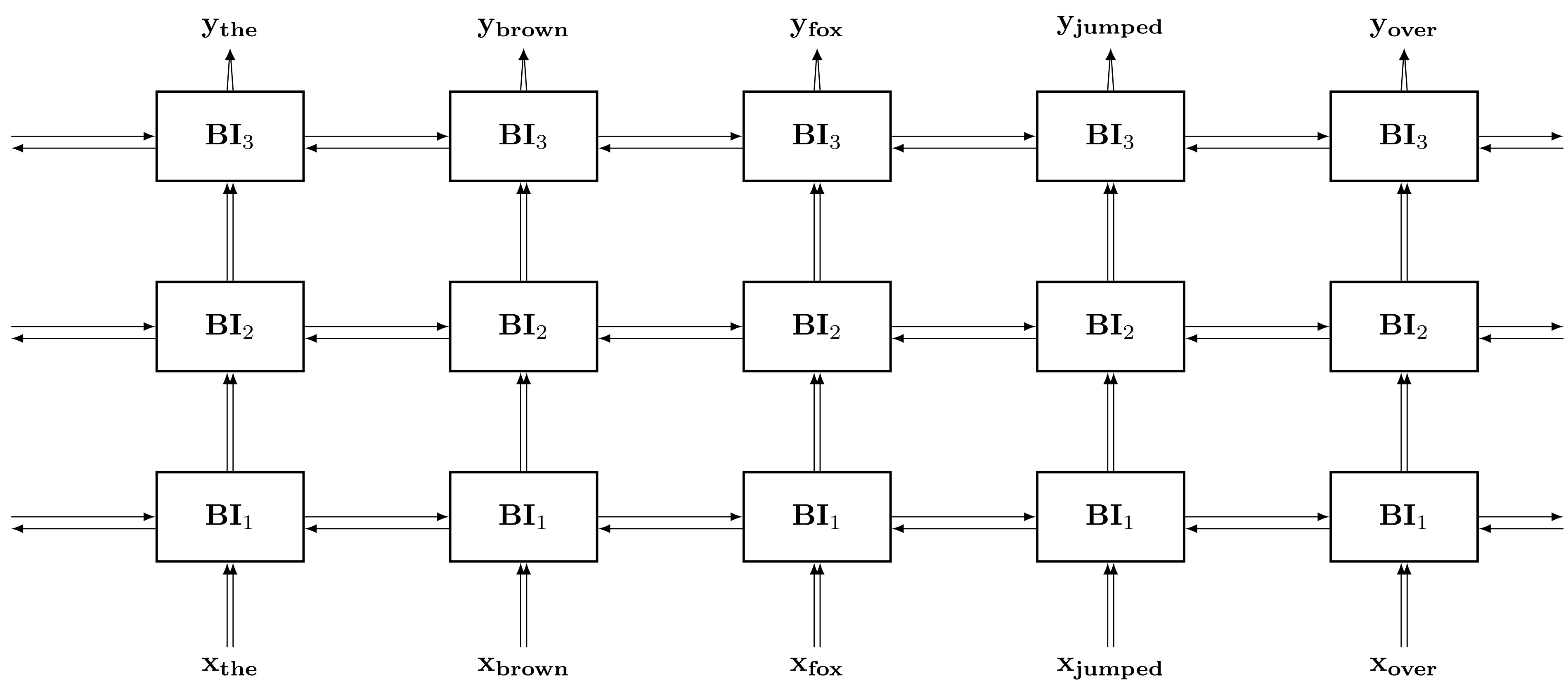
an infinite window around the word.



$$\begin{aligned} BiRNN(\mathbf{x_{1:7}}, 4) &= [\mathbf{y_4^F}; \mathbf{y_4^R}] \\ \mathbf{y_4^F} &= RNN_F(\mathbf{x_{1:4}}) \\ \mathbf{y_4^R} &= RNN_R(\mathbf{x_{7:4}}) \end{aligned}$$

One RNN runs left to right.
Another runs right to left.
Encode **both future and history** of a word.

Deep BI-RNNs



BI-RNN can also be stacked

(Deep) BI-RNNs

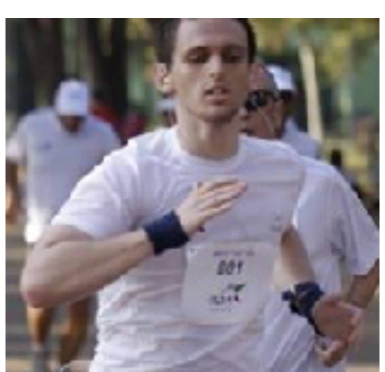
- provide an "infinite" window around a focus word.
- learn to extract what's important.
- easy to train!
- very effective for sequence tagging.
- Great as feature extractors!

Parsing with LSTMs, Take 2

Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations

Eliyahu Kiperwasser
Computer Science Department
Bar-Ilan University
Ramat-Gan, Israel
elikip@gmail.com

Yoav Goldberg
Computer Science Department
Bar-Ilan University
Ramat-Gan, Israel
yoav.goldberg@gmail.com



Parsing Background

There are two main frameworks for parsing:

- Graph-based:
  - Global inference
  - Score factorized over parts
  - There are first, second & third order parsers.
- Transition-based:
  - Greedy local inference
  - Score relies on current configuration, which is dependent on all previous transitions

Parsing Background

There are two main frameworks for parsing:

- Graph-based:
  - Global inference
  - Score factorized over parts
  - There are first, second & third order parsers.
- Transition-based:
  - Greedy local inference
  - Score relies on current configuration, which is dependent on all previous transitions

Structured Prediction Recipe

$$predict(x) = \arg\max_{y \in \mathcal{Y}(x)} \sum_{p \in y} score(\phi(p))$$

- Decompose structure to local factors.
- Assign a score to each factor.
- Structure score = sum of local scores.
- Look for highest scoring structure.

Graph-based Parsing

dobj

nsubj det prep nsubj

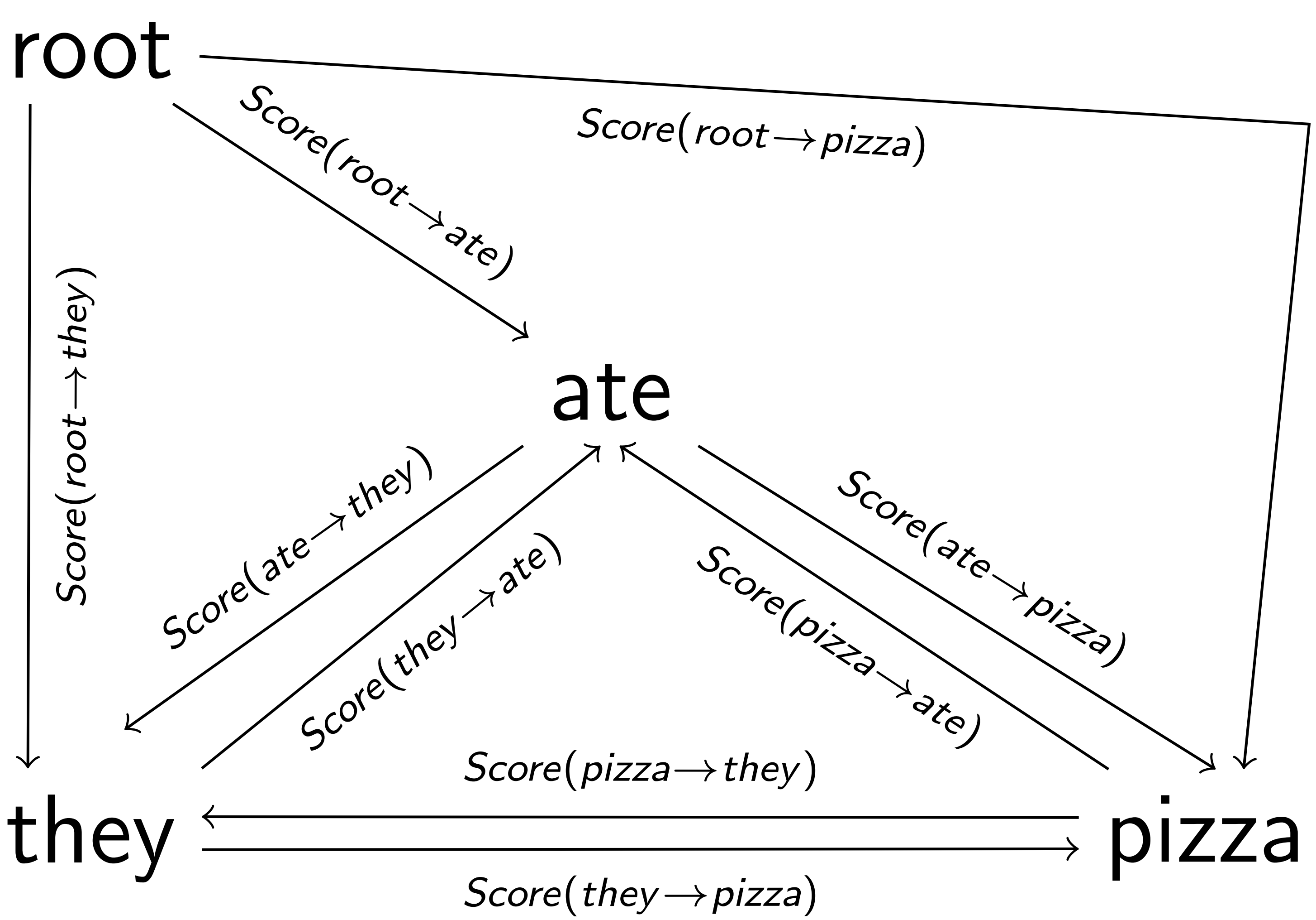
Score(They ate the pizza with anchovies) =

Score(They ate) + *Score*(ate pizza) + *Score*(the pizza) +

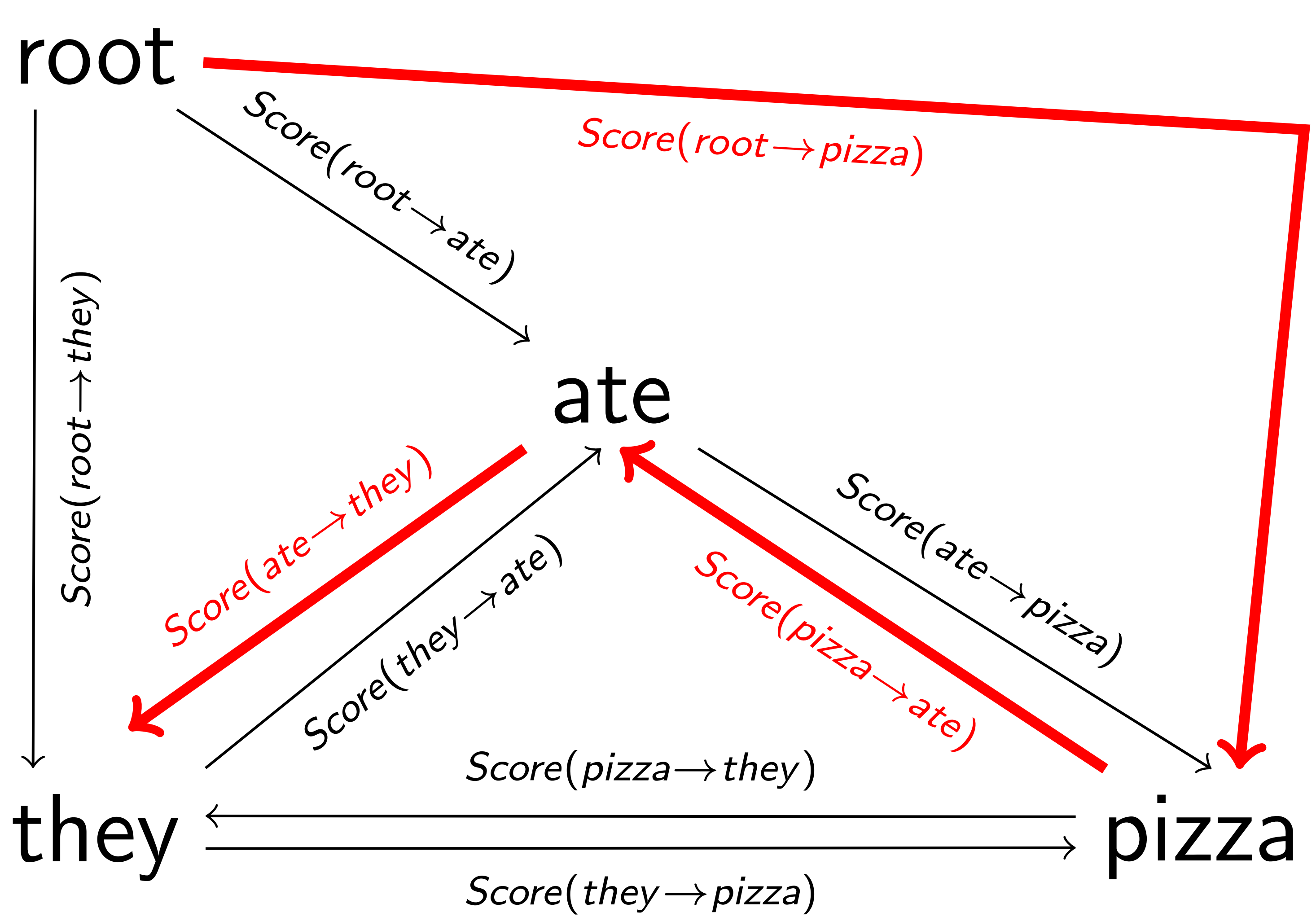
Score(pizza with) + *Score*(with anchovies)

Graph-based Parsing (Inference)

Input Sentence: "They ate pizza"



Graph-based Parsing (Inference)



Spanning tree with maximal score

Structured Prediction Recipe

$$predict(x) = \arg\max_{y \in \mathcal{Y}(x)} \sum_{p \in y} score(\color{red}{\phi(p)})$$

- *feature function* extracts useful signals from parts.
- most work goes into this component.

Arc Score Function

$$Score(\text{modifier} \curvearrowleft \text{head}) = ?$$

Arc Score Function

$$Score(\ \text{modifier} \quad \text{head}\ ) \ = \ F(\ \phi(modifier, head; sentence)\ )$$

- Similar story for transition-based parser
- **The choice of features is very important**

First-order features
(from Ryan McDonald's PhD thesis)

- Words and POS of Head and Mod.
- Words and POS of neighbors of Head and Mod.
- POS between Head and Modifier.
- Distance between Head and Modifier.
- Direction between Head and Modifier.
- Many, many **combination features.**

First-order features

(from Ryan McDonald's PhD thesis)

a)

Basic Uni-gram Features
x_i-word, x_i-pos
x_i-word
x_i-pos
x_j-word, x_j-pos
x_j-word
x_j-pos

b)

Basic Bi-gram Features
x_i-word, x_i-pos, x_j-word, x_j-pos
x_i-pos, x_j-word, x_j-pos
x_i-word, x_j-word, x_j-pos
x_i-word, x_i-pos, x_j-pos
x_i-word, x_i-pos, x_j-word
x_i-word, x_j-word
x_i-pos, x_j-pos

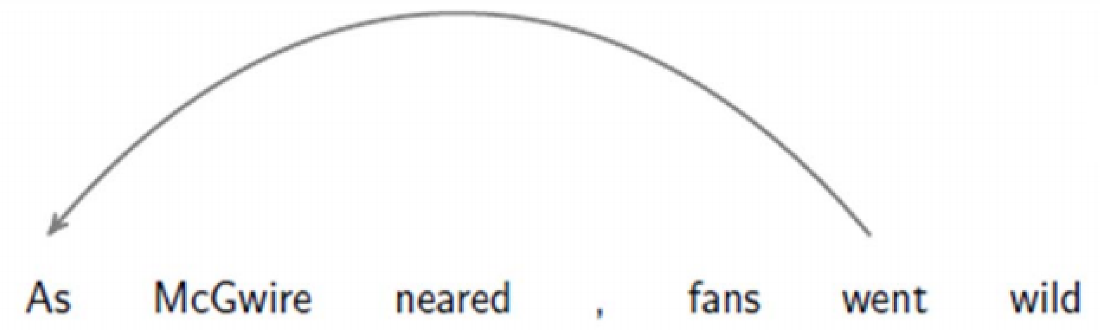
c)

In Between POS Features
x_i-pos, b-pos, x_j-pos
Surrounding Word POS Features
x_i-pos, x_i-pos+1, x_j-pos-1, x_j-pos
x_i-pos-1, x_i-pos, x_j-pos-1, x_j-pos
x_i-pos, x_i-pos+1, x_j-pos, x_j-pos+1
x_i-pos-1, x_i-pos, x_j-pos, x_j-pos+1

Table 3.1: Features used by system, $f(i, j)$, where x_i is the head and x_j the modifier in the dependency relation. x_i-word: word of head in dependency edge. x_j-word: word of modifier. x_i-pos: POS of head. x_j-pos: POS of modifier. x_i-pos+1: POS to the right of head in sentence. x_i-pos-1: POS to the left of head. x_j-pos+1: POS to the right of modifier. x_j-pos-1: POS to the left of modifier. b-pos: POS of a word in between head and modifier.

Manual Feature Templates

Core Features + Feature Combinations



[went]	[VBD]	[As]	[ADP]	[went]
[VERB]	[As]	[IN]	[went, VBD]	[As, ADP]
[went, As]	[VBD, ADP]	[went, VERB]	[As, IN]	[went, As]
[VERB, IN]	[VBD, As, ADP]	[went, As, ADP]	[went, VBD, ADP]	[went, VBD, As]
[ADJ, *, ADP]	[VBD, *, ADP]	[VBD, ADJ, ADP]	[VBD, ADJ, *]	[NNS, *, ADP]
[NNS, VBD, ADP]	[NNS, VBD, *]	[ADJ, ADP, NNP]	[VBD, ADP, NNP]	[VBD, ADJ, NNP]
[NNS, ADP, NNP]	[NNS, VBD, NNP]	[went, left, 5]	[VBD, left, 5]	[As, left, 5]
[ADP, left, 5]	[VERB, As, IN]	[went, As, IN]	[went, VERB, IN]	[went, VERB, As]
[JJ, *, IN]	[VERB, *, IN]	[VERB, JJ, IN]	[VERB, JJ, *]	[NOUN, *, IN]
[NOUN, VERB, IN]	[NOUN, VERB, *]	[JJ, IN, NOUN]	[VERB, IN, NOUN]	[VERB, JJ, NOUN]
[NOUN, IN, NOUN]	[NOUN, VERB, NOUN]	[went, left, 5]	[VERB, left, 5]	[As, left, 5]
[IN, left, 5]	[went, VBD, As, ADP]	[VBD, ADJ, *, ADP]	[NNS, VBD, *, ADP]	[VBD, ADJ, ADP, NNP]
[NNS, VBD, ADP, NNP]	[went, VBD, left, 5]	[As, ADP, left, 5]	[went, As, left, 5]	[VBD, ADP, left, 5]
[went, VERB, As, IN]	[VERB, JJ, *, IN]	[NOUN, VERB, *, IN]	[VERB, JJ, IN, NOUN]	[NOUN, VERB, IN, NOUN]
[went, VERB, left, 5]	[As, IN, left, 5]	[went, As, left, 5]	[VERB, IN, left, 5]	[VBD, As, ADP, left, 5]
[went, As, ADP, left, 5]	[went, VBD, ADP, left, 5]	[went, VBD, As, left, 5]	[ADJ, *, ADP, left, 5]	[VBD, *, ADP, left, 5]
[VBD, ADJ, ADP, left, 5]	[VBD, ADJ, *, left, 5]	[NNS, *, ADP, left, 5]	[NNS, VBD, ADP, left, 5]	[NNS, VBD, *, left, 5]
[ADJ, ADP, NNP, left, 5]	[VBD, ADP, NNP, left, 5]	[VBD, ADJ, NNP, left, 5]	[NNS, ADP, NNP, left, 5]	[NNS, VBD, NNP, left, 5]
[VERB, As, IN, left, 5]	[went, As, IN, left, 5]	[went, VERB, IN, left, 5]	[went, VERB, As, left, 5]	[JJ, *, IN, left, 5]
[VERB, *, IN, left, 5]	[VERB, JJ, IN, left, 5]	[VERB, JJ, *, left, 5]	[NOUN, *, IN, left, 5]	[NOUN, VERB, IN, left, 5]

Example from slides of Rush and Petrov (2012)

Core Features + ~~Feature Combinations~~

replace feature combinations with non-linear learner

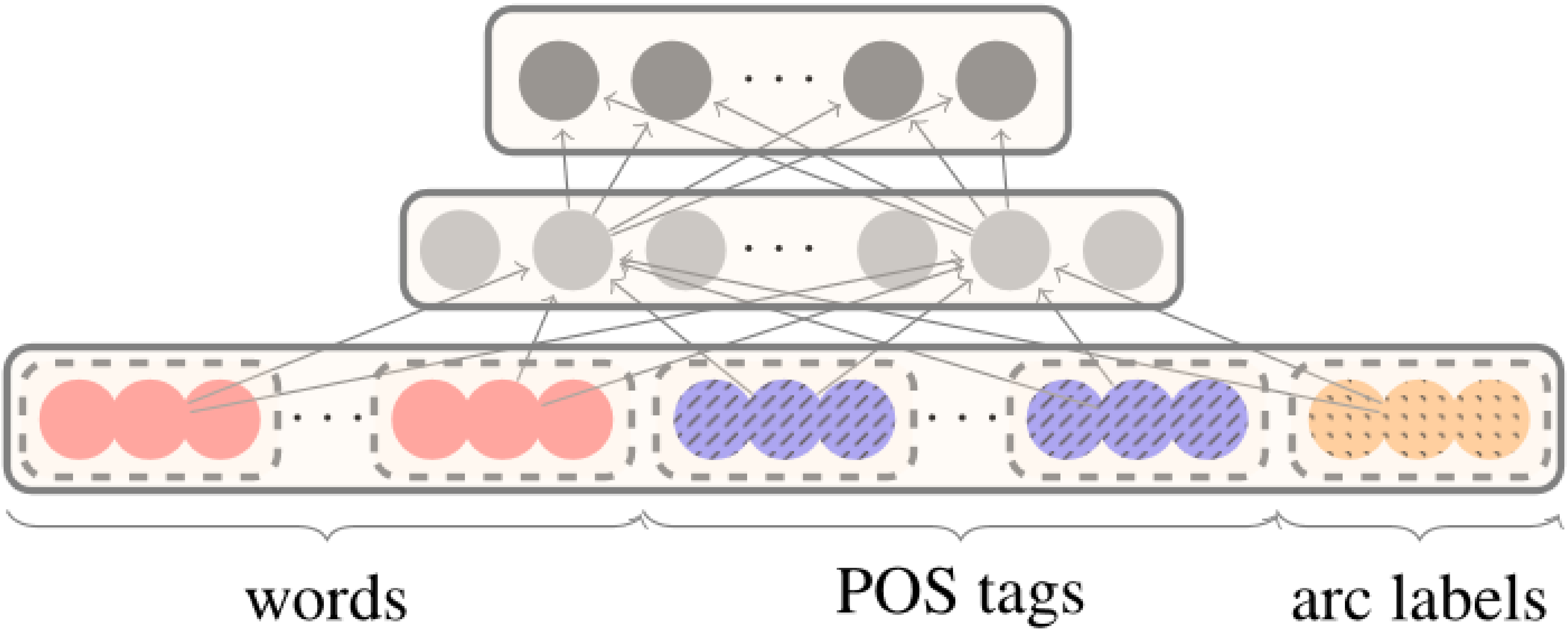


Figure from Chen and Manning (2014)
Similiar approach in Pei et al, Weiss et al, Andor et al

Core Features + Non-Linear Classifier

replace feature combinations with non-linear learner

but still need to define good features.

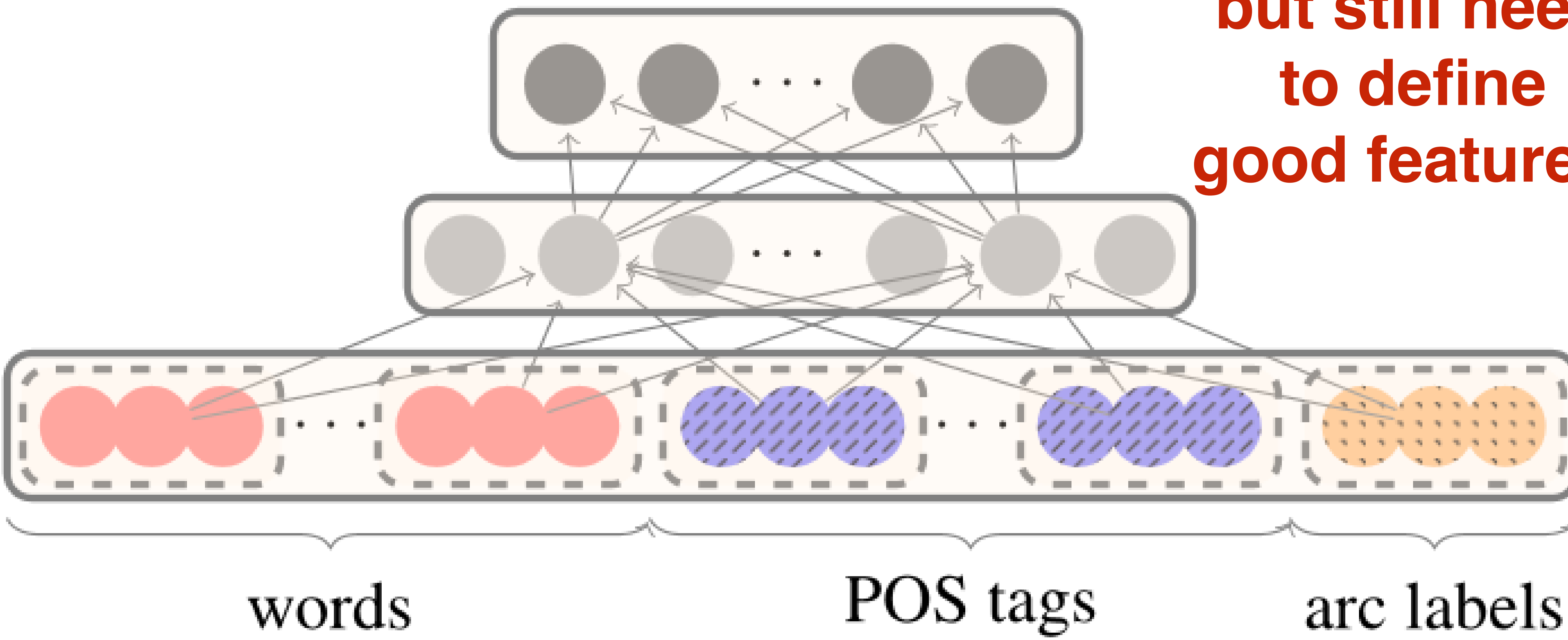
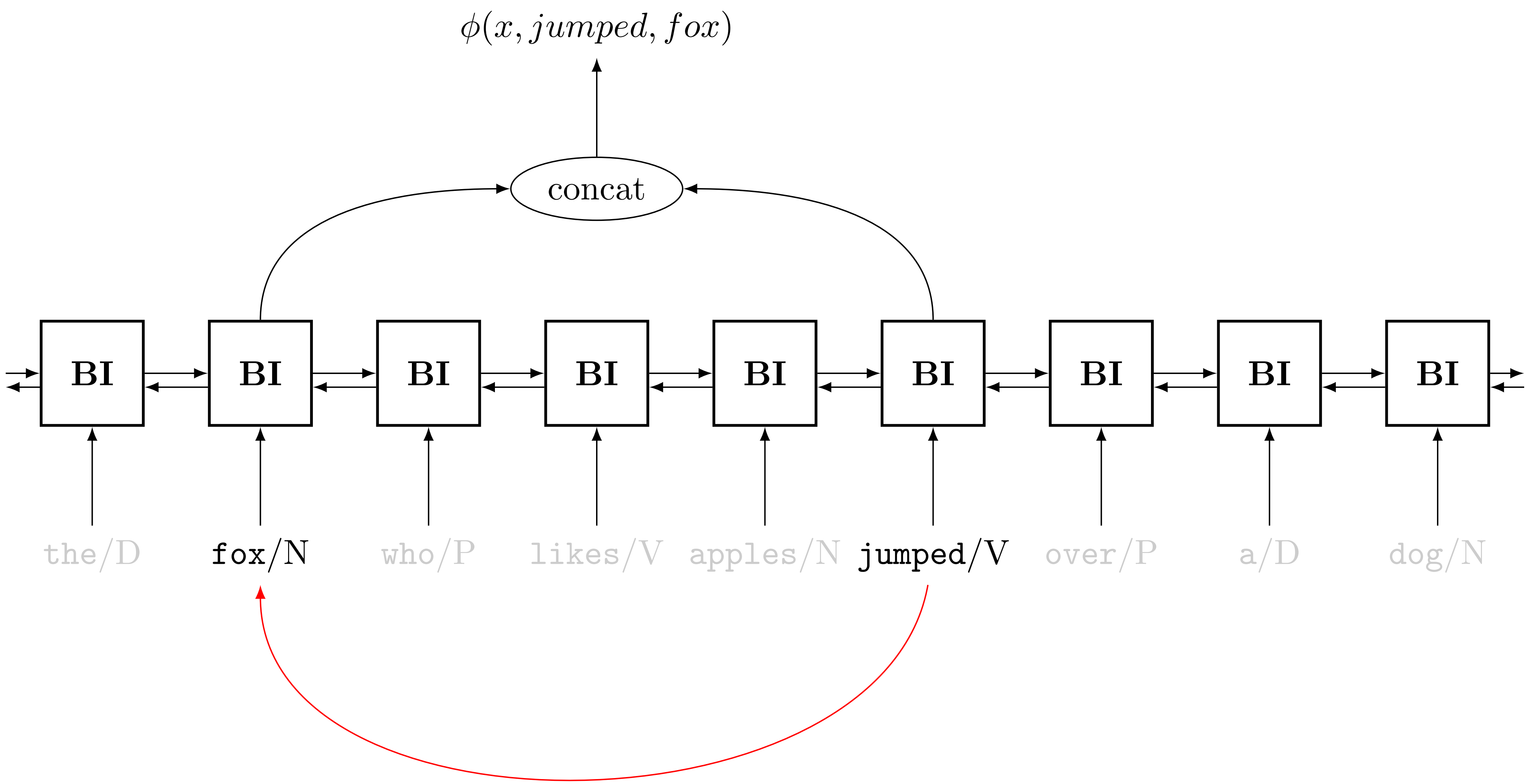


Figure from Chen and Manning (2014)
Similiar approach in Pei et al, Weiss et al, Andor et al

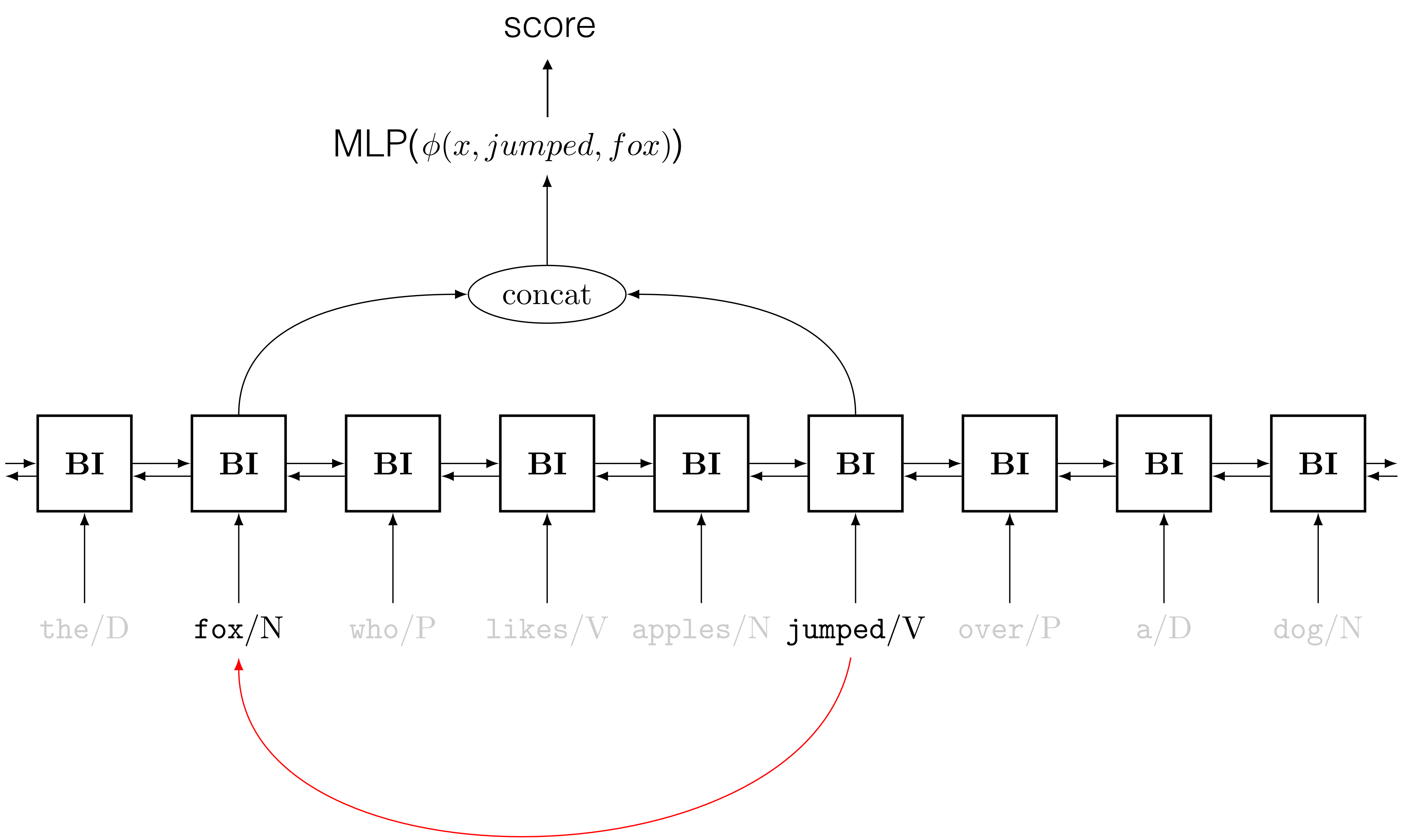
Our take on it

Let's just use a Bidirectional LSTM



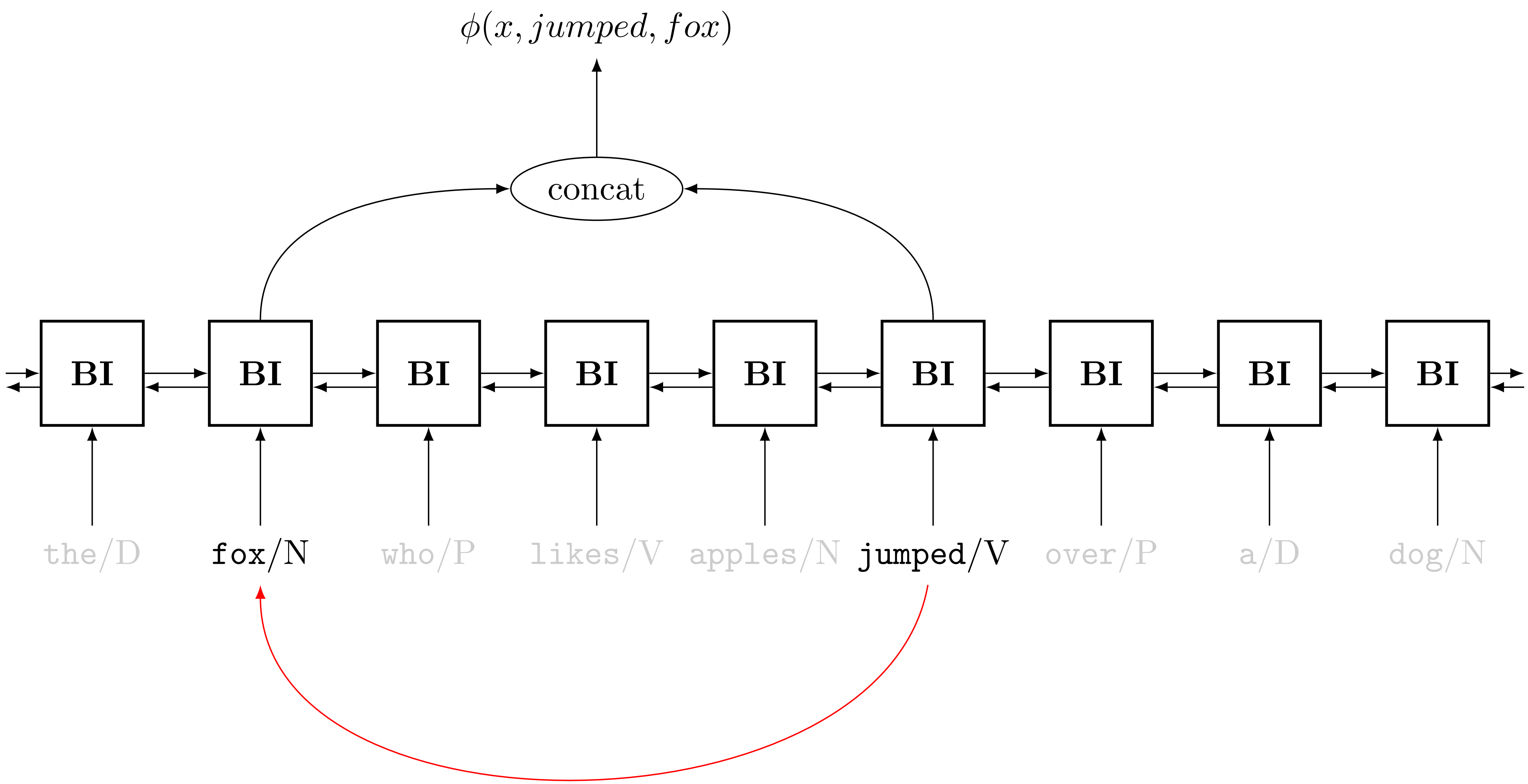
$$score(h, m, x) = MLP(\phi(x, h, m))$$

$$\phi(x, h, m) = [BIRNN(x, h); BIRNN(x, m)]$$



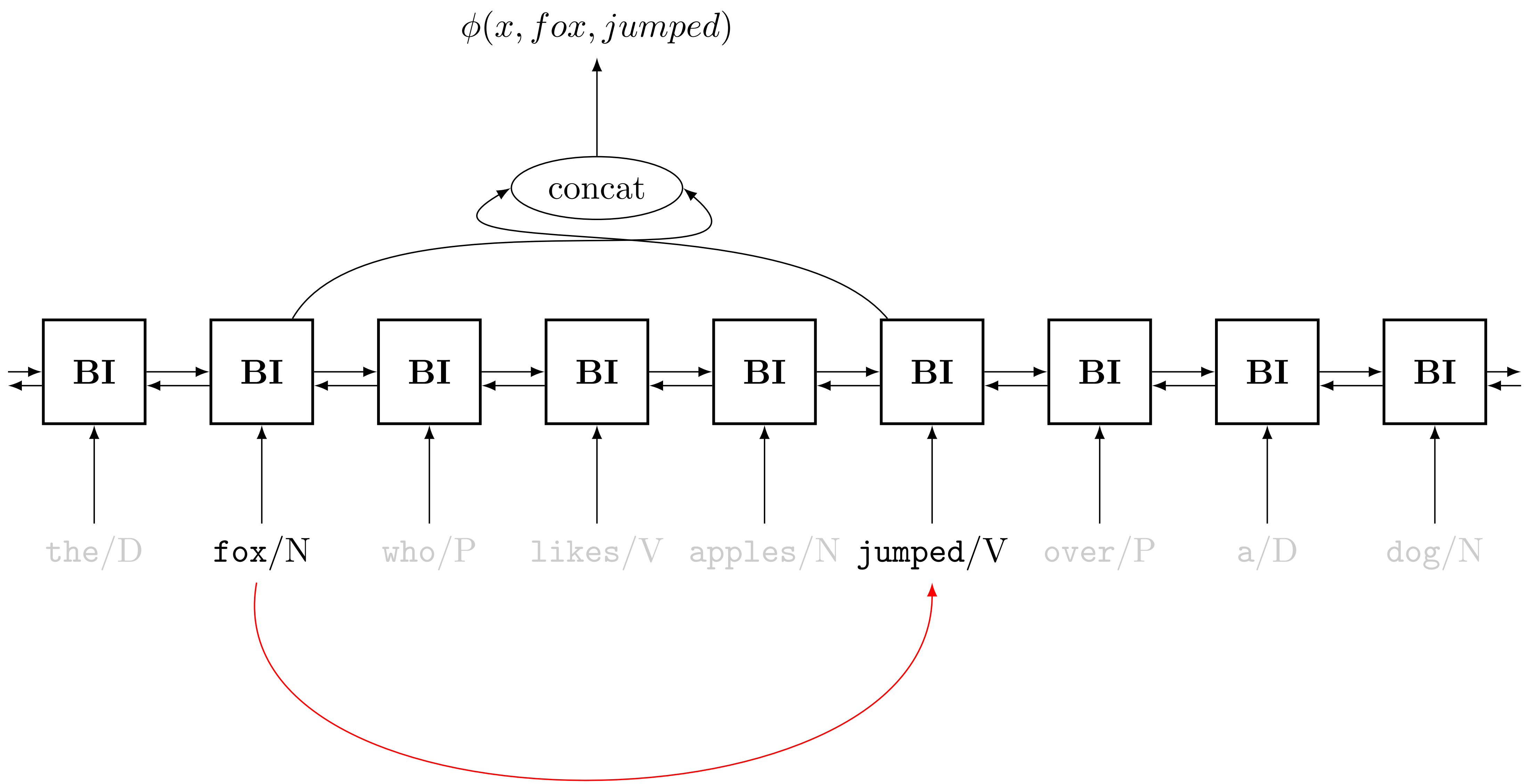
$$score(h, m, x) = MLP(\phi(x, h, m))$$

$$\phi(x, h, m) = [BIRNN(x, h); BIRNN(x, m)]$$



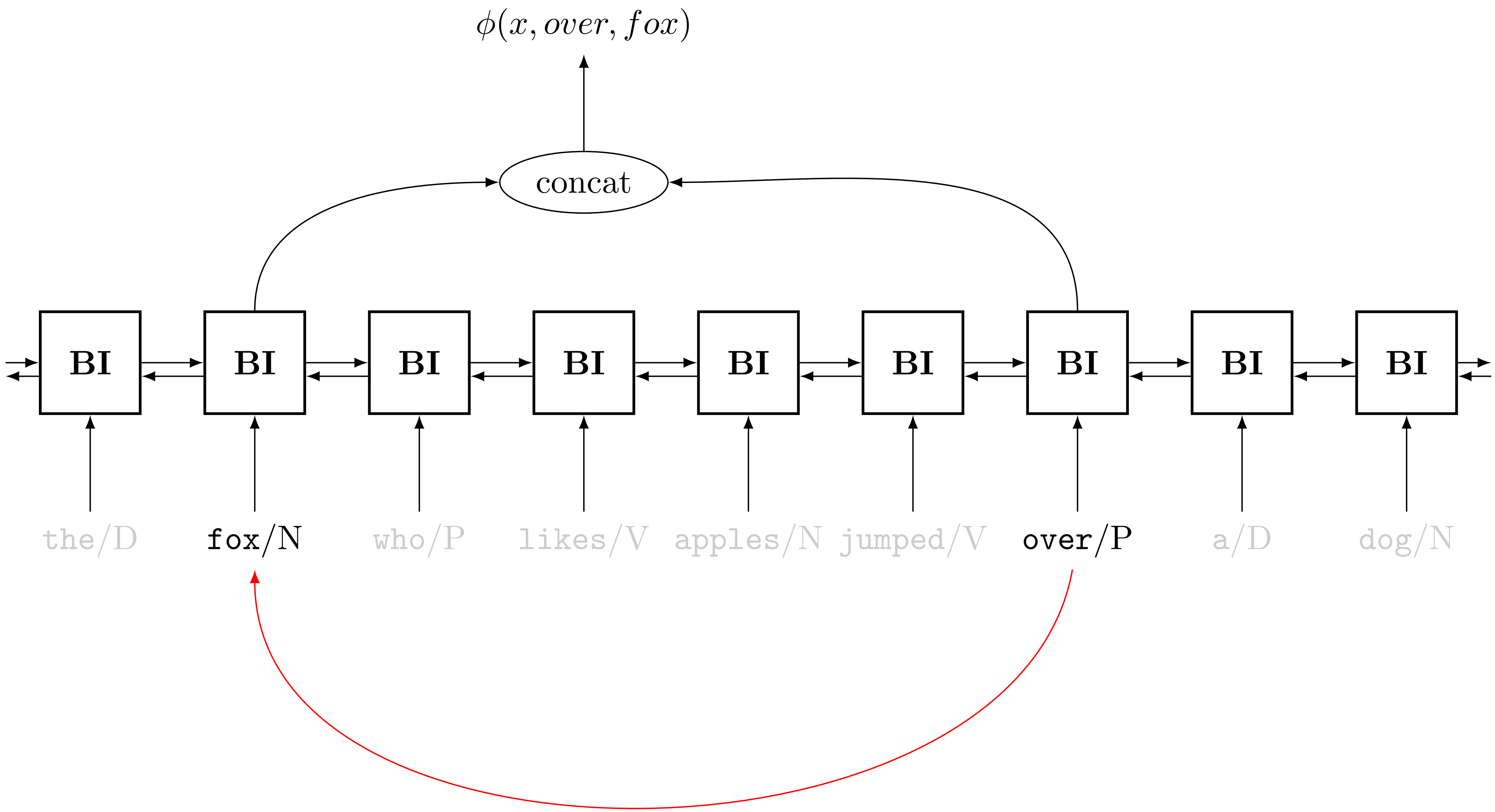
$$score(h, m, x) = MLP(\phi(x, h, m))$$

$$\phi(x, h, m) = [BIRNN(x, h); BIRNN(x, m)]$$



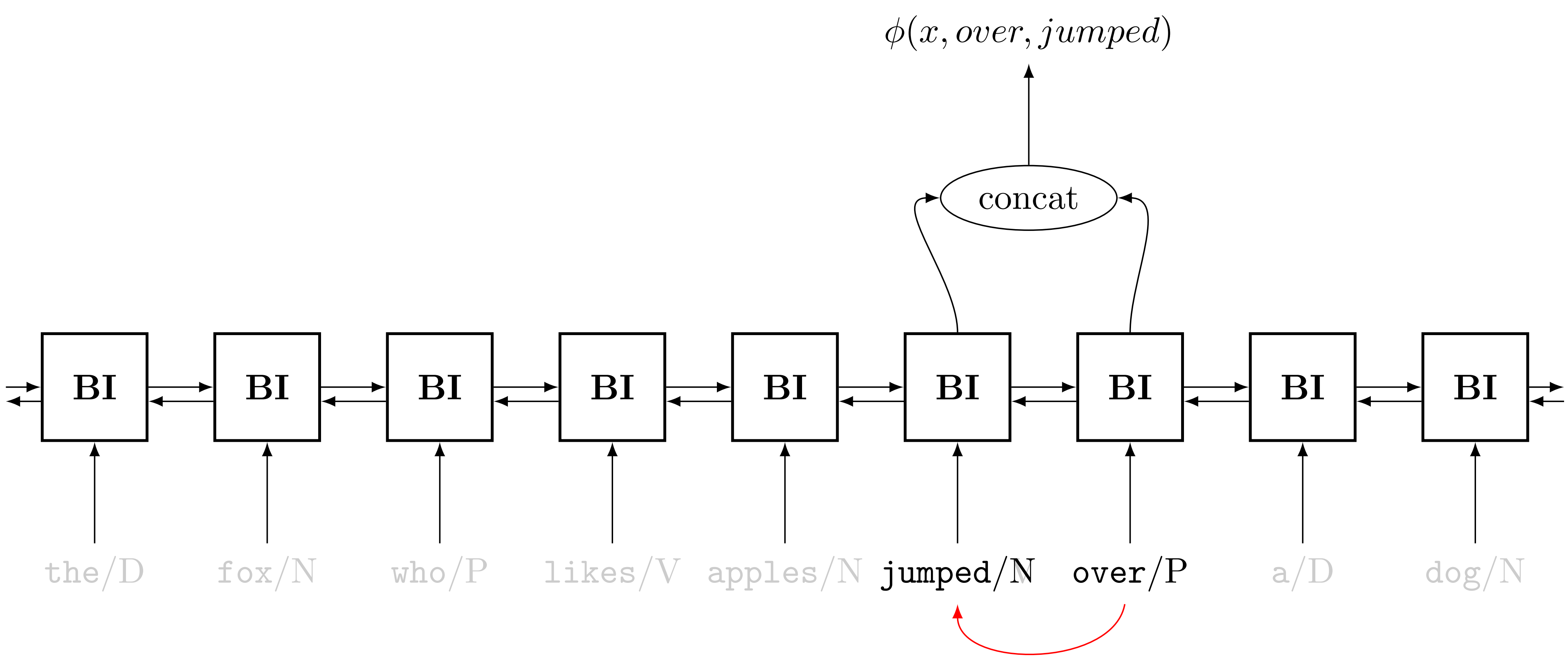
$$score(h, m, x) = MLP(\phi(x, h, m))$$

$$\phi(x, h, m) = [BIRNN(x, h); BIRNN(x, m)]$$



$$score(h, m, x) = MLP(\phi(x, h, m))$$

$$\phi(x, h, m) = [BIRNN(x, h); BIRNN(x, m)]$$

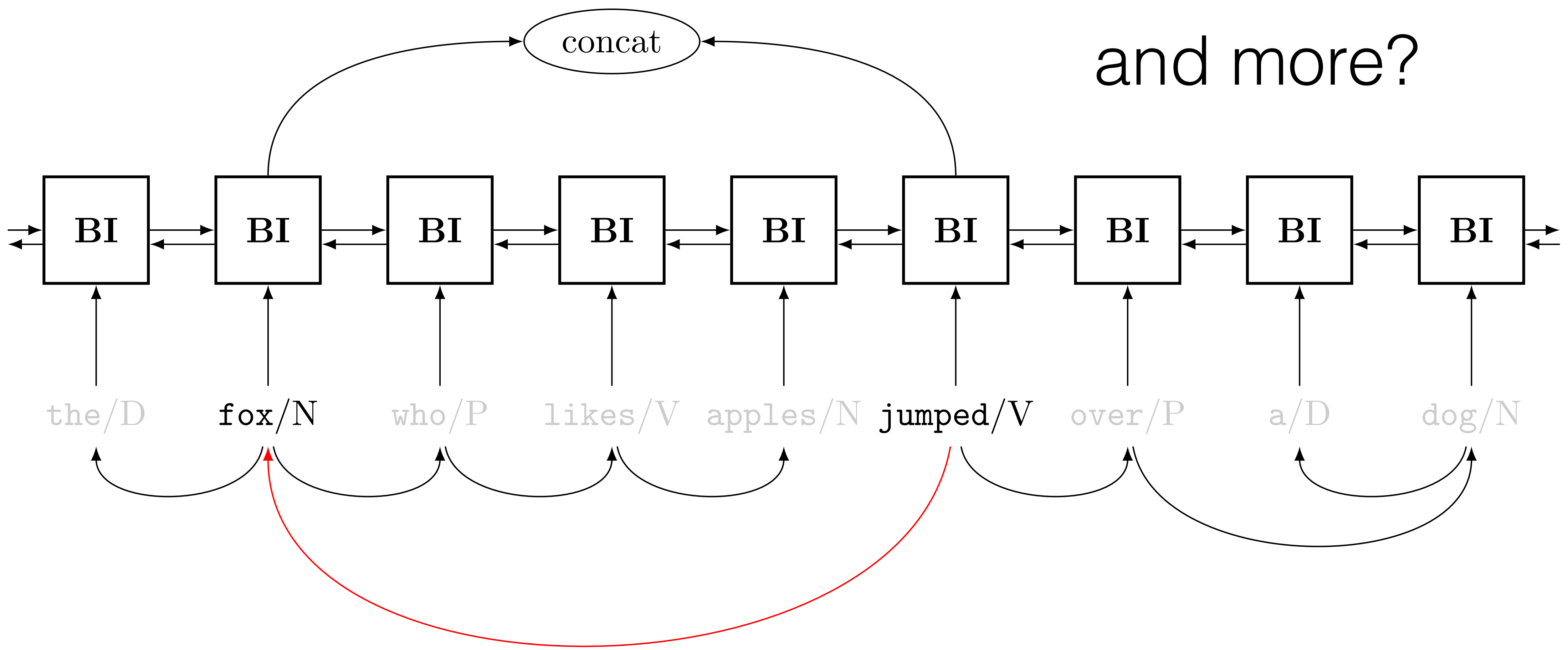


$$score(h, m, x) = MLP(\phi(x, h, m))$$

$$\phi(x, h, m) = [BIRNN(x, h); BIRNN(x, m)]$$

the two BI-RNN vectors give us:

infinite window around head
infinite window around mod
distance between head and mod
content between head and mod
and more?



$$score(h, m, x) = MLP(\phi(x, h, m))$$

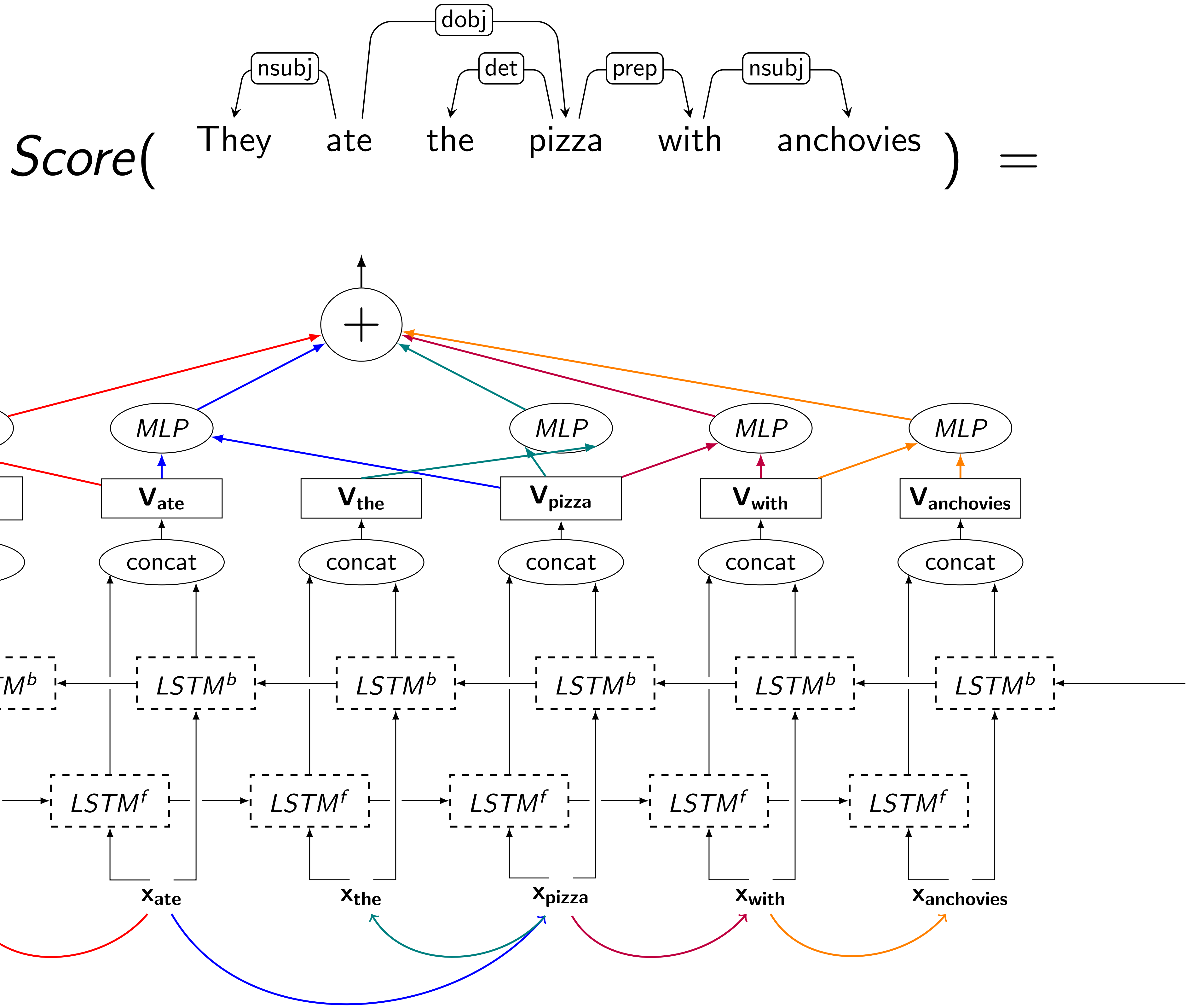
$$\phi(x, h, m) = [BIRNN(x, h); BIRNN(x, m)]$$

Arc Score (Intuition)

- The BiLSTM encoding of a word holds information about its attachment preferences
- The score is dependent on the BiLSTM encoding which in turn depends on the entire sentence
- Therefore, the score function focused on a specific arc is considering also the entire sentence attachement preferences

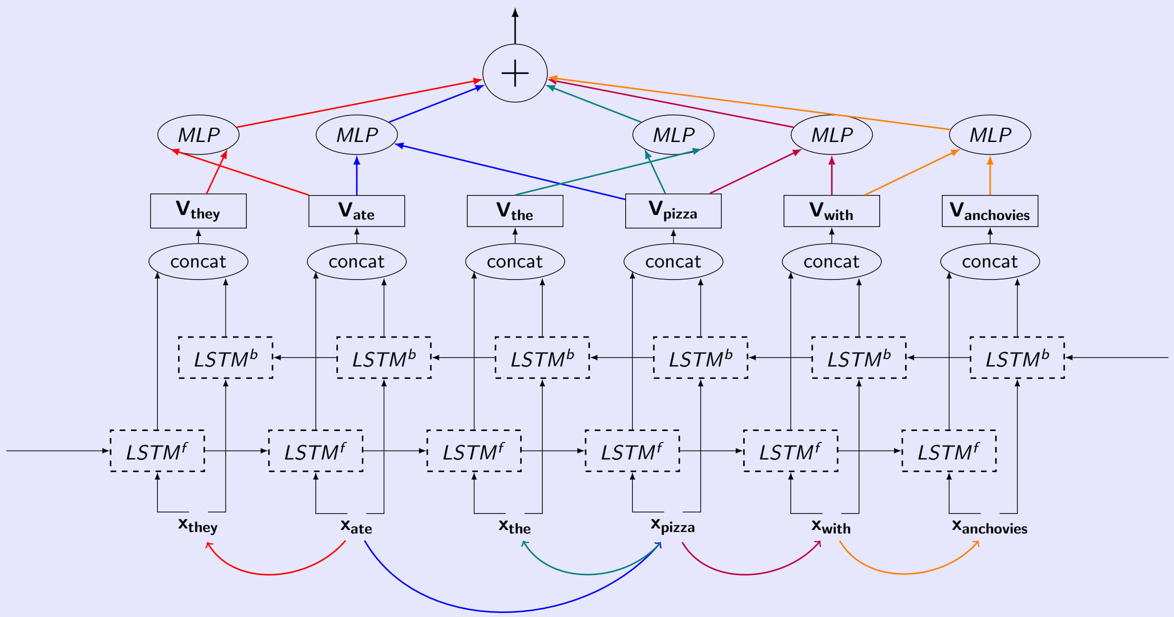
Tree Score

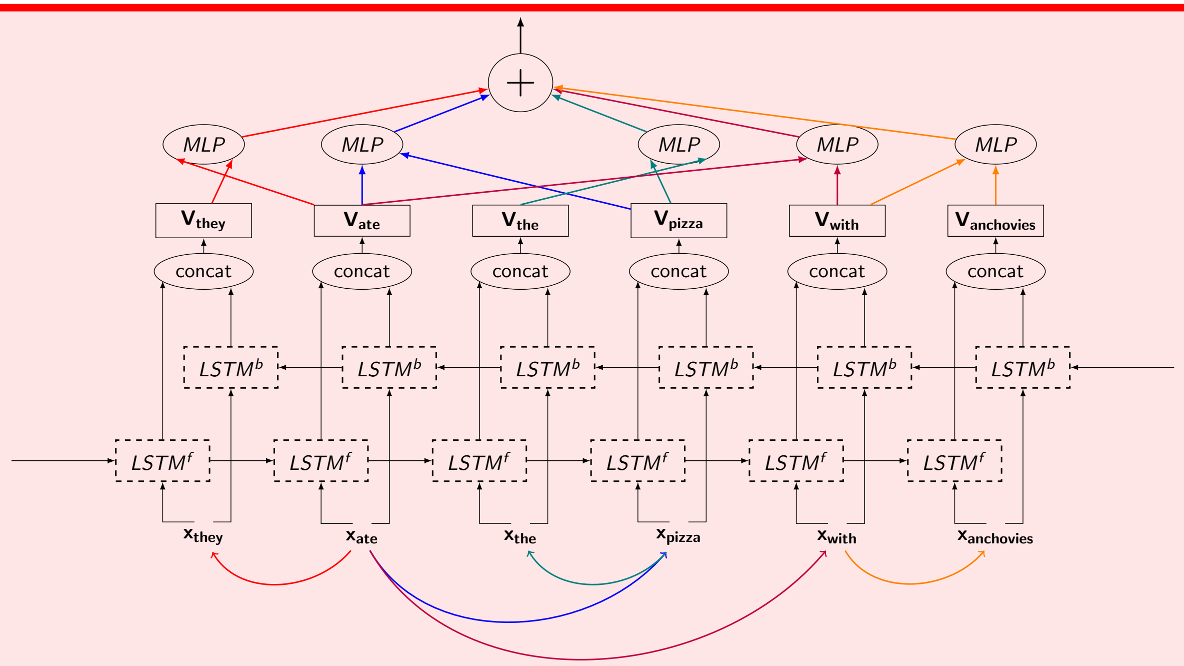

Yum!

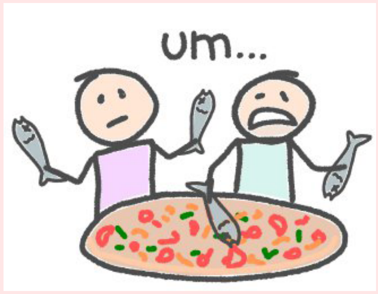

Score(They ate the pizza with anchovies) =
nsubj
dobj
det
prep
nsubj
MLP
MLP
MLP
MLP
MLP
$\mathbf{v}_{\text{they}}$
$\mathbf{v}_{\text{ate}}$
$\mathbf{v}_{\text{the}}$
$\mathbf{v}_{\text{pizza}}$
$\mathbf{v}_{\text{with}}$
$\mathbf{v}_{\text{anchovies}}$
concat
$LSTM^b$
$LSTM^f$
$\mathbf{x}_{\text{they}}$
$\mathbf{x}_{\text{ate}}$
$\mathbf{x}_{\text{the}}$
$\mathbf{x}_{\text{pizza}}$
$\mathbf{x}_{\text{with}}$
$\mathbf{x}_{\text{anchovies}}$

Large Margin Objective

$$max(0, 1 - \boxed{} + \boxed{})$$







Training Objective

Gold tree should score a margin above all other trees

$$\sum_{(h,m)\in y} MLP(\phi(x,h,m)) \quad - \sum_{(h,m)\in y' \neq y} MLP(\phi(x,h,m)) > 1$$

$$\phi(x,h,m) = [BIRNN(x,h); BIRNN(x,m)]$$

Backdrop all the way back through the BI-LSTM

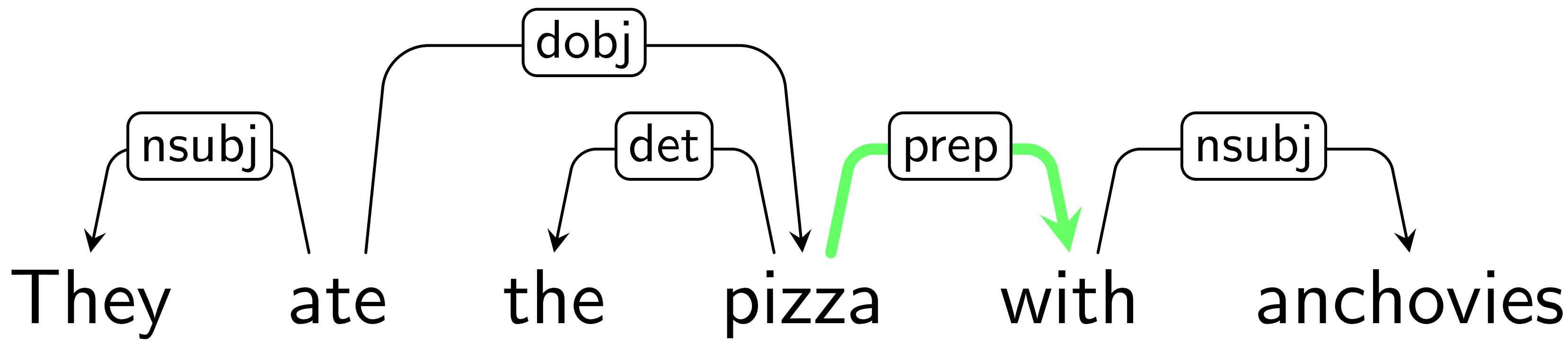
Graph-based Parsing (More Details)

- **Cost Augmentation**: Make non-gold attachments more attractive in training by adding a constant to their score
- **Multi-Task Learning**: Learning the label on the same BiLSTM representation helps both in terms of accuracy and performance.
- **For Speed**: Simple algebric "trick" reduces the number of matrix multiplication significantly.

Graph-based Parsing (More Details)

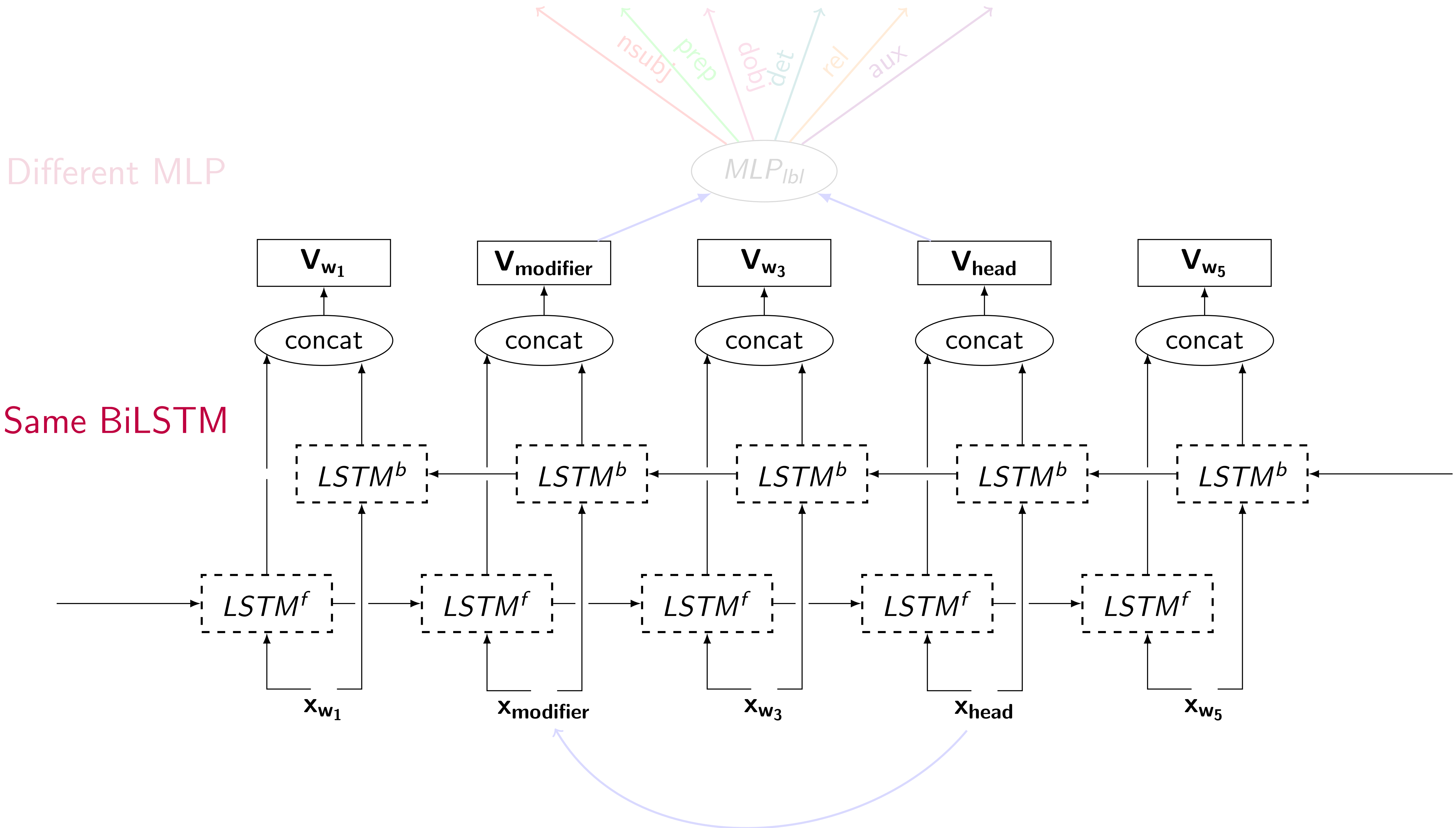
- **Cost Augmentation**: Make non-gold attachments more attractive in training by adding a constant to their score
- **Multi-Task Learning**: Learning the label on the same BiLSTM representation helps both in terms of accuracy and performance.
- **For Speed**: Simple algebric "trick" reduces the number of matrix multiplication significantly.

Arc Labels (Multi-Task Learning)



- The arc labels hold important additional syntactic information
- The labels contribute information useful for the unlabeled case too

Arc Labels (Multi-Task Learning)

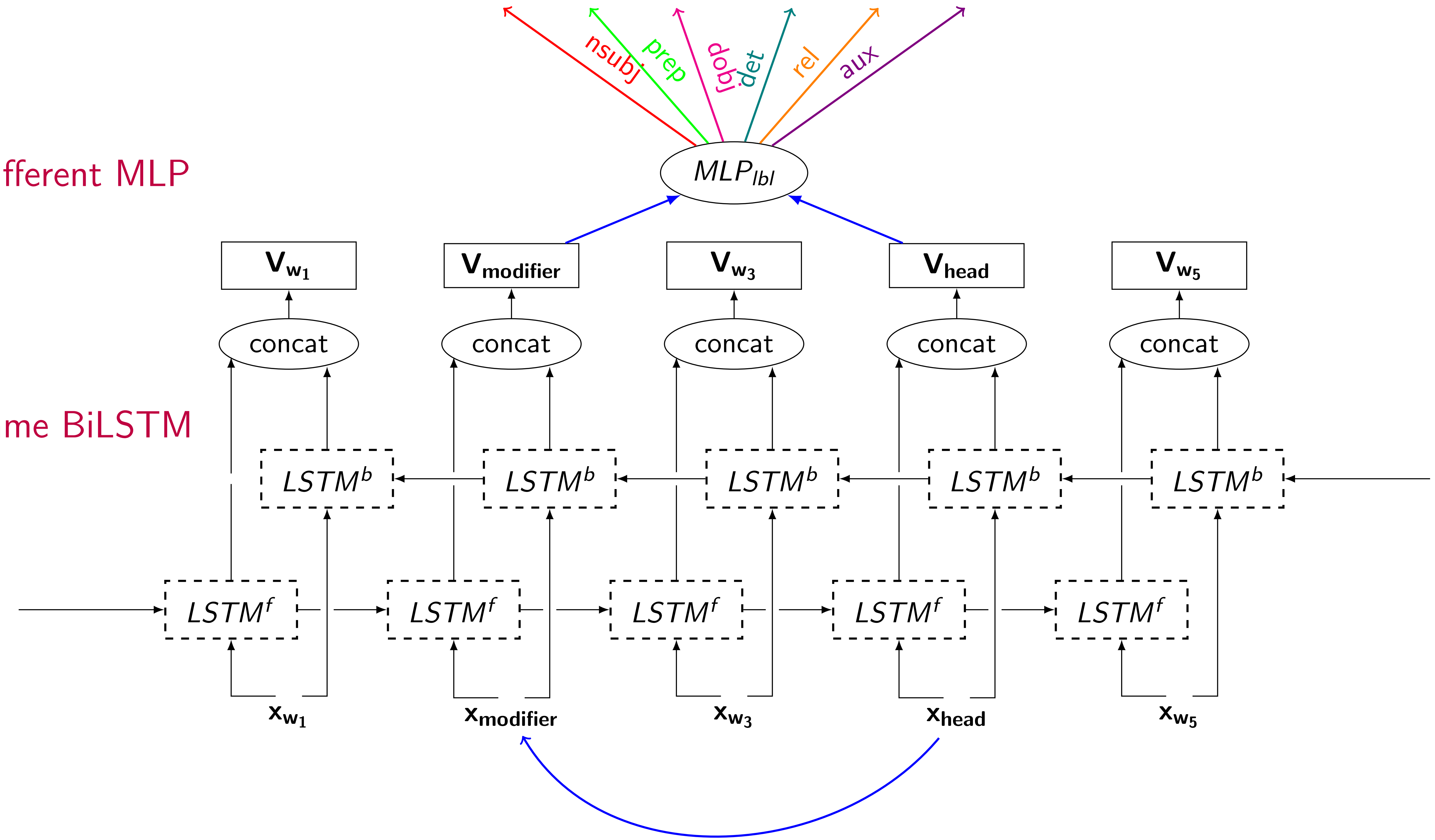


Enrich BiLSTM representation by learning labels

Arc Labels (Multi-Task Learning)

Different MLP

Same BiLSTM



Enrich BiLSTM representation by learning labels

In parsing time

- Run (deep) BI-LSTM over words+POS.
    - this gives us a vector $\mathbf{v_i}$ for each word.
- Compute scores for each arc (h,m) via $MLP([\mathbf{v_h};\mathbf{v_m}])$
- Decode using arc scores.

Graph-based Parsing

and this works:

93.2 UAS with two features,

first-order parser,

without external embeddings.

Graph-based Parsing

and this works:

93.2 UAS with two features,

first-order parser,

without external embeddings.

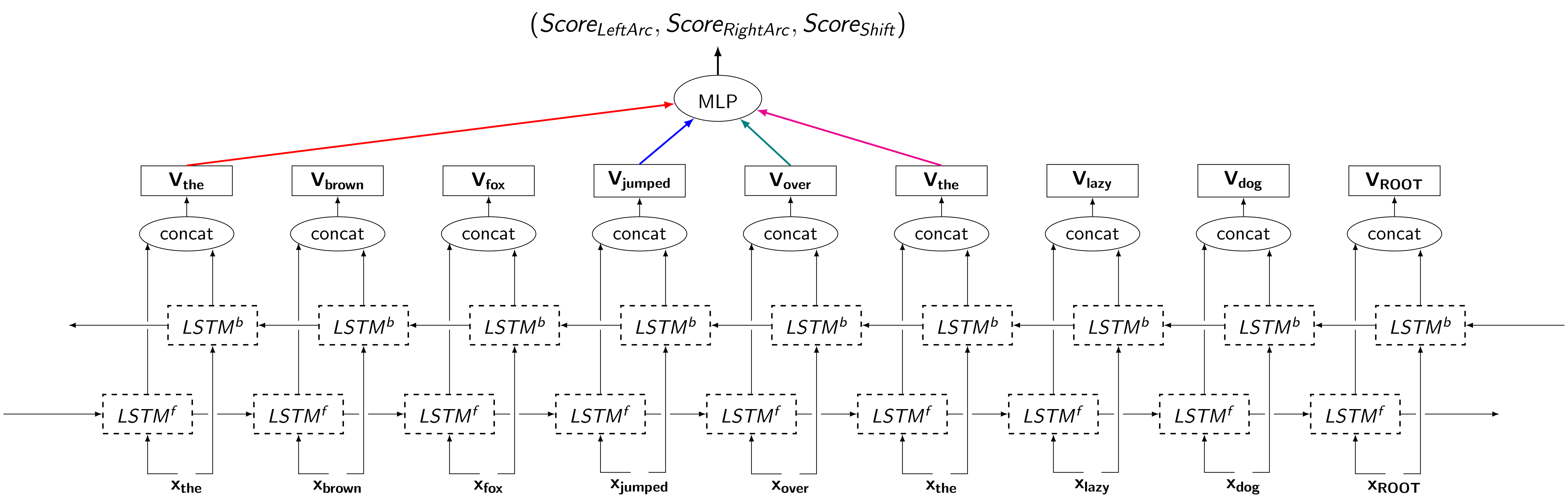
This is remarkably effective!

We can use same trick also for Transition based parsing

Transition-based Parsing (Oracle)

Configuration:

Scoring:



also worth noting:

Incremental Parsing with Minimal Features Using Bi-Directional LSTM

James Cross and **Liang Huang**
School of Electrical Engineering and Computer Science
Oregon State University
Corvallis, Oregon, USA
{crossj,liang.huang}@oregonstate.edu

Constituency Parsing
Transition-based

also worth noting:

Fast(er) Exact Decoding and Global Training for Transition-Based Dependency Parsing via a Minimal Feature Set

Tianze Shi
Cornell University
tianze@cs.cornell.edu

Liang Huang
Oregon State University
liang.huang.sh@gmail.com

Lillian Lee
Cornell University
llee@cs.cornell.edu

Dependency Parsing
Transition-based + Dynamic Programming

also worth noting:

Transition-Based Dependency Parsing with Stack Long Short-Term Memory

Chris Dyer[♣♠] **Miguel Ballesteros**[♢♠] **Wang Ling**[♠] **Austin Matthews**[♠] **Noah A. Smith**[♠]
[♣]Marianas Labs [♢]NLP Group, Pompeu Fabra University [♠]Carnegie Mellon University
chris@marianaslabs.com, miguel.ballesteros@upf.edu,
{lingwang,austinma,nasmith}@cs.cmu.edu

in retrospect
"Stack LSTM" parser is very similar to the biLSTM
(but does have extra compositionality)

But let's get back to the 1st-order Graph Parser

1st order Decomposition is Incredibly Naive

nsubj dobj det prep nsubj

Score(They ate the pizza with anchovies) =

Score(They ate) + *Score*(ate pizza) + *Score*(the pizza) +

Score(pizza with) + *Score*(with anchovies)

And yet...

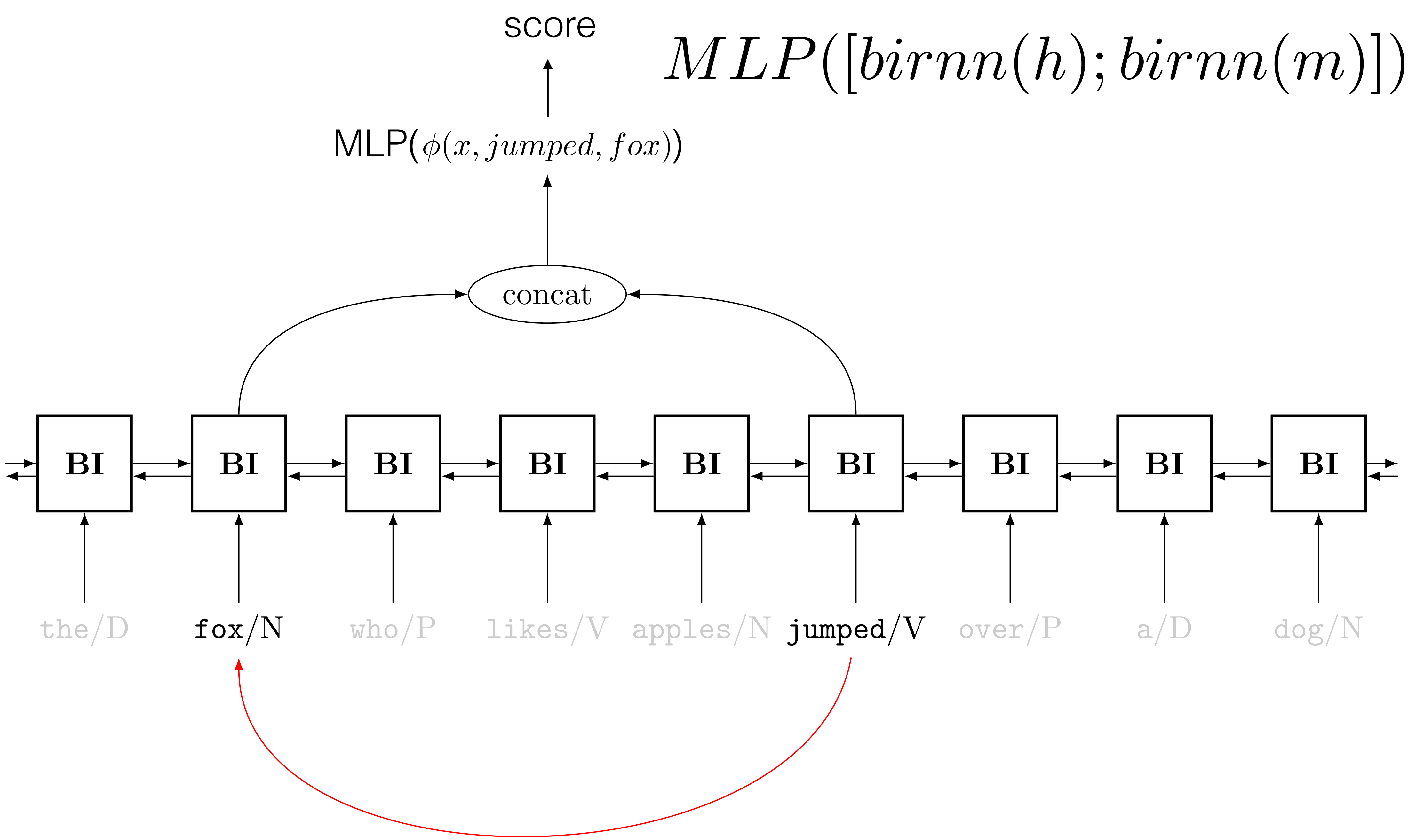
RBG Parser (Lei et al, 2014), 1st order:	91.7 UAS
TurboPasrer (Martins et al, 2013), 3rd order:	93.1 UAS
BiLSTM (K&G, 2016), 1st order:	93.2 UAS
BiLSTM (K&G, 2016), + embeddings:	92.7 UAS
BiLSTM (K&G, 2016), + emb, bug fix:	94.0 UAS
Dozat and Manning 2017:	

DEEP BIAFFINE ATTENTION FOR NEURAL DEPENDENCY PARSING

Timothy Dozat
Stanford University
tdozat@stanford.edu

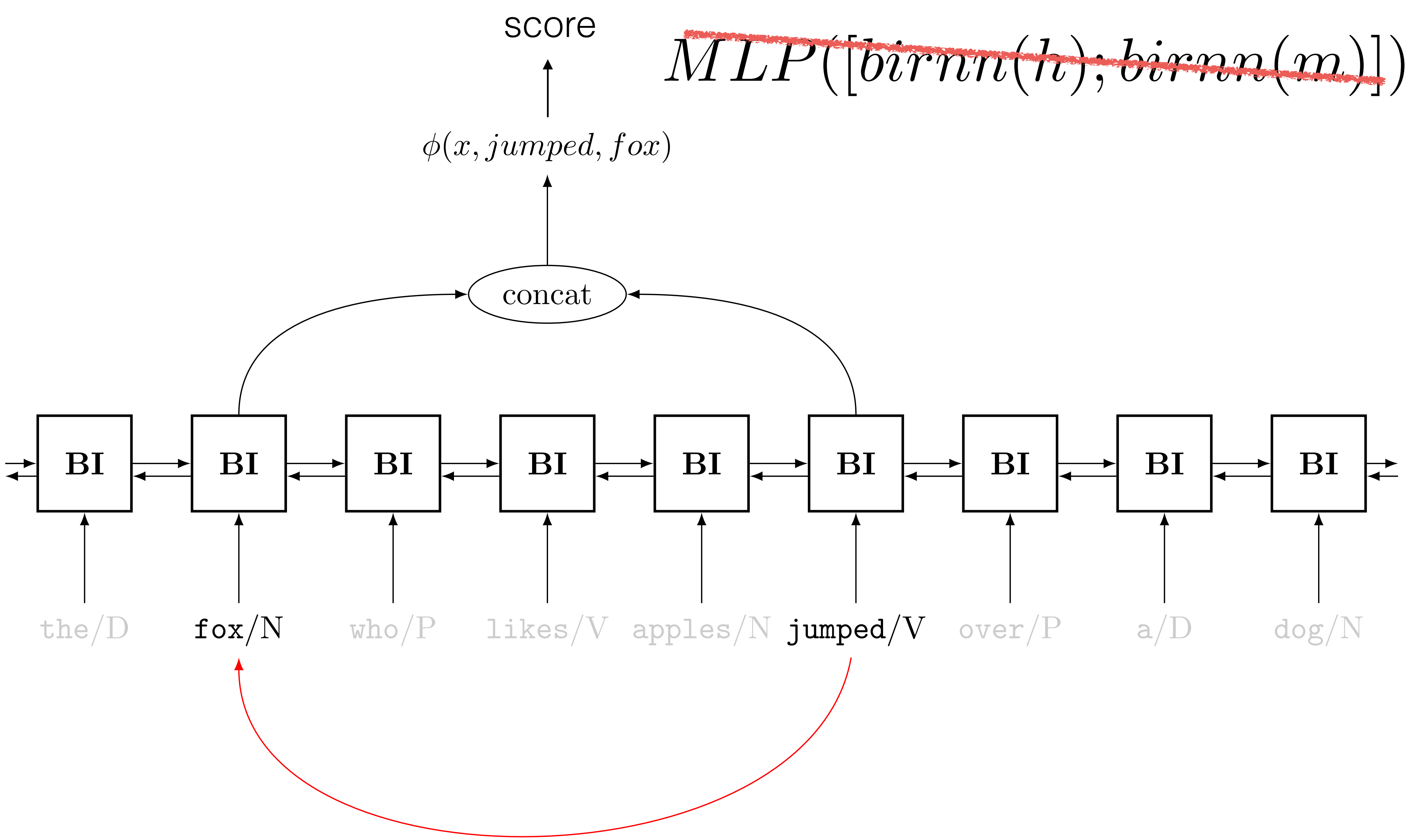
Christopher D. Manning
Stanford University
manning@stanford.edu

$$MLP([birnn(h); birnn(m)])$$



K&G 2016

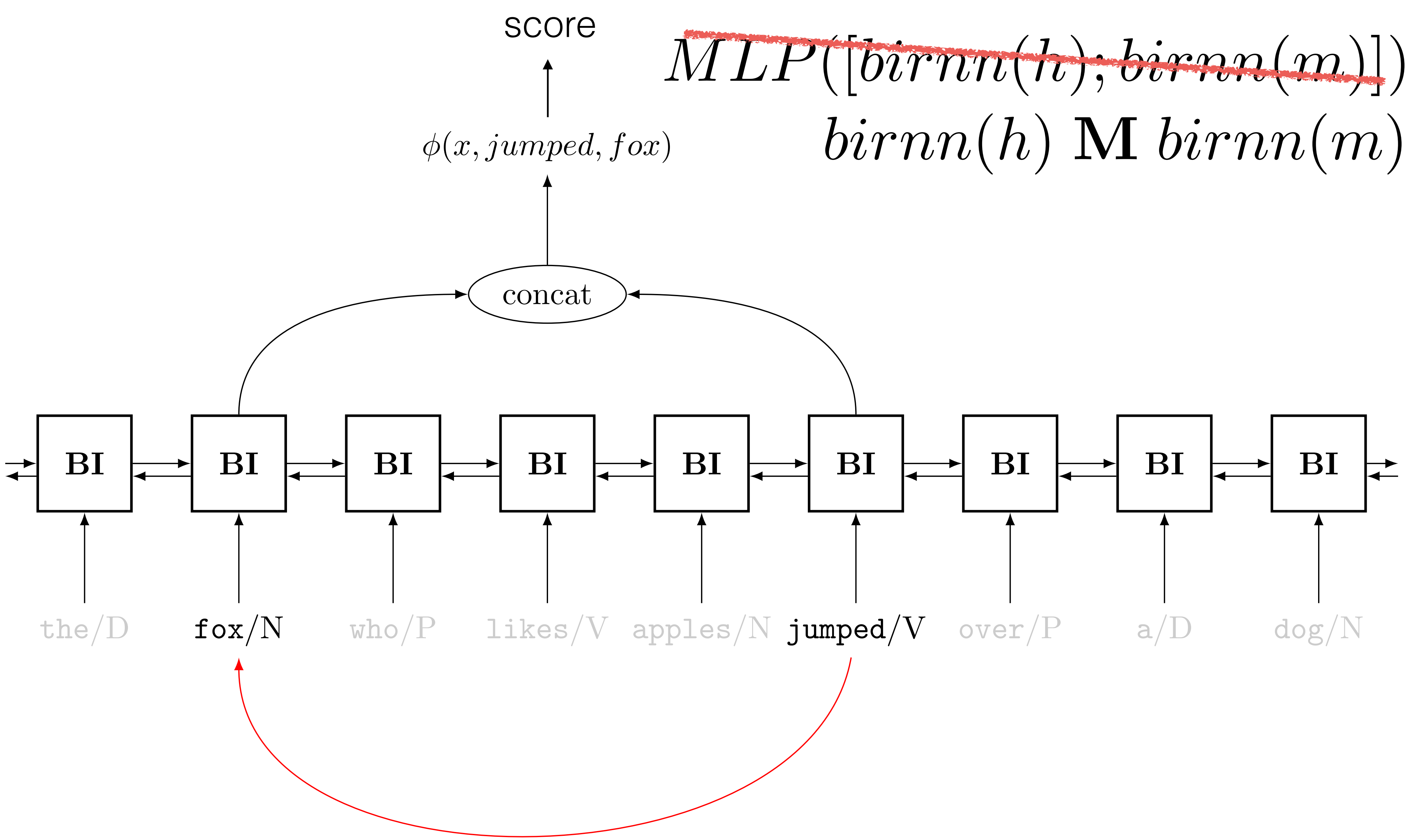
$\cancel{MLP([birnn(h);birnn(m)])}$



Dozat and Manning, 2017

$$\cancel{MLP([birnn(h); birnn(m)])}$$

$$birnn(h)\ \mathbf{M}\ birnn(m)$$



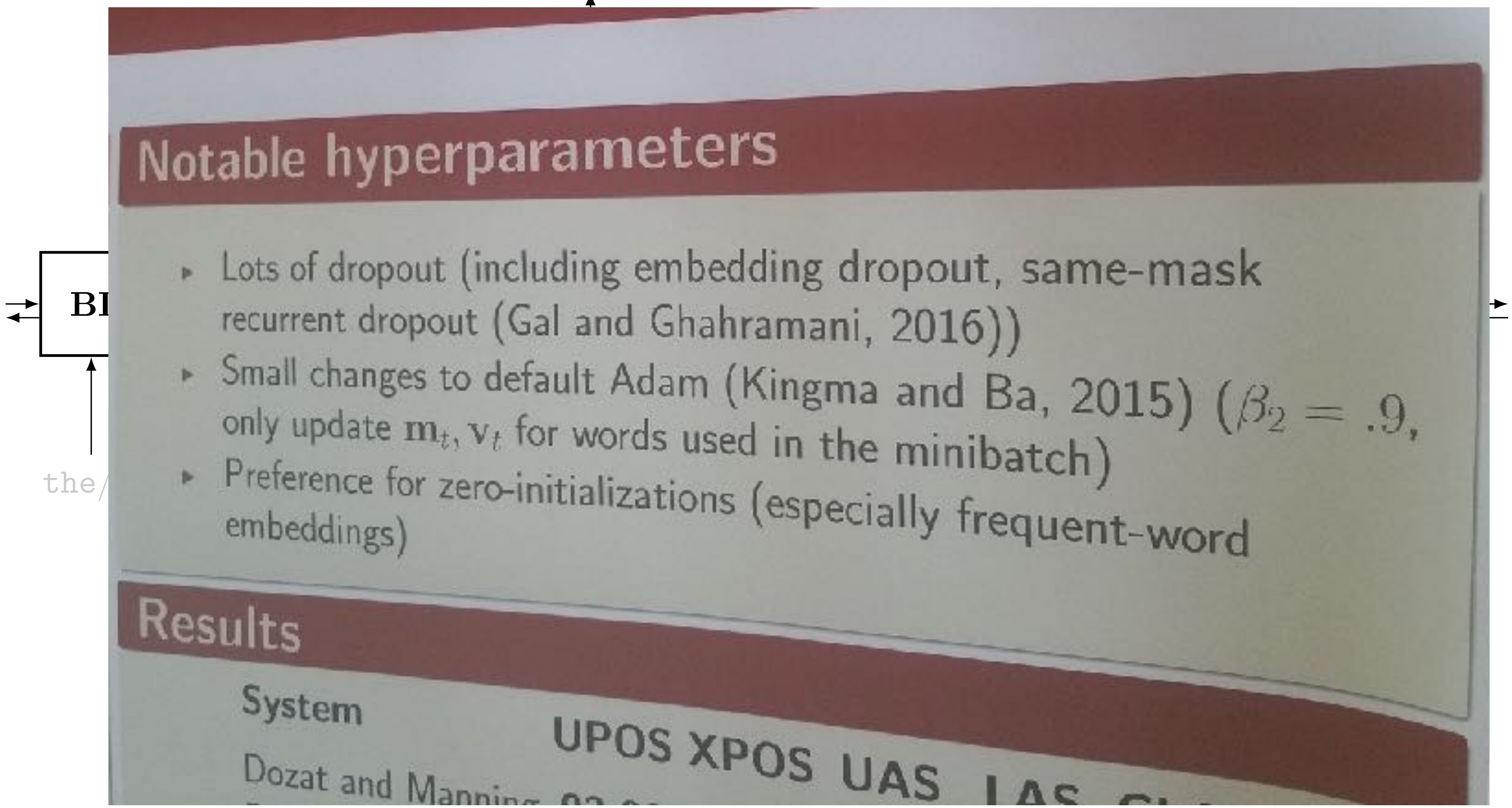
Dozat and Manning, 2017

score

$\phi(x, jumped, fox)$

$\cancel{MLP([birnn(h); birnn(m)])}$

$birnn(h)\ \mathbf{M}\ birnn(m)$



Dozat and Manning, 2017

And yet...

RBG Parser (Lei et al, 2014), 1st order:	91.7 UAS
TurboPasrer (Martins et al, 2013), 3rd order:	93.1 UAS
BiLSTM (K&G, 2016), 1st order:	93.2 UAS
BiLSTM (K&G, 2016), + embeddings:	92.7 UAS
BiLSTM (K&G, 2016), + emb, bug fix:	94.0 UAS
Dozat and Manning 2017: (BiLSTM. First Order)	**95.7 UAS**

CoNLL 2017 Shared Task

CoNLL 2017 Shared Task

Results: Unlabeled Attachment Score (UAS)

All treebanks

1.	Stanford (Stanford)	software1	81.30
2.	C2L2 (Ithaca)	software5	80.35
3.	IMS (Stuttgart)	software2	79.90
4.	HIT-SCIR (Harbin)	software4	77.81
5.	LATTICE (Paris)	software7	76.75
6.	NAIST SATO (Nara)	software1	76.35
7.	UParse (Edinburgh)	software1	75.49
8.	Koç University (İstanbul)	software3	75.44
9.	ÚFAL – UDPipe 1.2 (Praha)	software1	75.39
10.	Orange – Deskiñ (Lannion)	software1	75.11
11.	RACAI (București)	software1	74.67
12.	LyS-FASTPARSE (A Coruña)	software5	74.42

CoNLL 2017 Shared Task

Results: Unlabeled Attachment Score (UAS)

Dozat and Manning
biLSTM + graph + tuning
(first-order features)

All treebanks

1. Stanford (Stanford)	software1	81.30
2. C2L2 (Ithaca)	software5	80.35
3. IMS (Stuttgart)	software2	79.90
4. HIT-SCIR (Harbin)	software4	77.81
5. LATTICE (Paris)	software7	76.75
6. NAIST SATO (Nara)	software1	76.35
7. UParse (Edinburgh)	software1	75.49
8. Koç University (İstanbul)	software3	75.44
9. ÚFAL – UDPipe 1.2 (Praha)	software1	75.39
10. Orange – Deskiñ (Lannion)	software1	75.11
11. RACAI (București)	software1	74.67
12. LyS-FASTPARSE (A Coruña)	software5	74.42

CoNLL 2017 Shared Task

Results: Unlabeled Attachment Score (UAS)

All treebanks

Dozat and Manning
biLSTM + graph + tuning
(first-order features)

model of **Shi, Huang and Lee**
biLSTM + transition + DP
(first-order features)

1.	Stanford (Stanford)	software1	81.30
2.	C2L2 (Ithaca)	software5	80.35
3.	IMS (Stuttgart)		79.90
4.	HIT-SCIR (Harbi		77.81
5.	LATTICE (Paris)		76.75
6.	NAIST SATO (Nar		76.35
7.	UParse (Edinbur		75.49
8.	Koç University (İstanbul)	software3	75.44
9.	ÚFAL – UDPipe 1.2 (Praha)	software1	75.39
10.	Orange – Deskiñ (Lannion)	software1	75.11
11.	RACAI (București)	software1	74.67
12.	LyS-FASTPARSE (A Coruña)	software5	74.42

CoNLL 2017 Shared Task

Results: Unlabeled Attachment Score (UAS)

All treebanks

Dozat and Manning
biLSTM + graph + tuning
(first-order features)

1.	Stanford (Stanford)	software1	81.30
2.	C2L2 (Ithaca)	software5	80.35
3.	IMS (Stuttgart)		79.90
4.	HIT-SCIR (Harbi		77.81
5.	LATTICE (Paris)		76.75
6.	NAIST SATO (Nar		76.35
7.	UParse (Edinbur		75.49
8.	Koç University (İstanbul)	software3	75.44
9.	ÚFAL – UDPipe 1.2 (Praha)	software1	75.39
10.			5.11
11.			4.67
12.	LyS-FASTPARSE (A Coruña)	software5	74.42

model of **Shi, Huang and Lee**
biLSTM + transition + DP
(first-order features)

(both used also character-level LSTMs for words)

The best parsers
in the world today
are based on
1st-order decomposition
over a BiLSTM

The best parsers
in the world today
are based on
1st-order decomposition
over a BiLSTM

I find this remarkable

Take home questions

- Why does it work?
- What is encoded in these vectors?
- Where does it fail?
- How can we improve? (in an interesting way)?

  morphology? pre-training? multi-tasking? composition?

Take home questions

- Why does it work?
- What is encoded in these vectors?
- Where does it fail?
- How can we improve? (in an interesting way)?

morphology? pre-training? multi-tasking? composition?

thanks for listening!