Semantic Parsing

Natalie Parde

UIC CS 421

What is semantic parsing?

- The process of extracting semantic structure or meaning from natural language input
  - What are the semantic **dependencies** present in the language sample?
  - How do elements in the language sample relate to one another **logically**?
  - What **semantic roles** are filled in the language sample?

Most popular semantic parsing task: dependency parsing

- Automatically determining **directed grammatical and semantic relationships** between words
  - **Semantic:** Focused on **meaning**
- This information is useful for many NLP applications, including:
  - Coreference resolution
  - Question answering
  - Information extraction

How are dependency grammars different from CFGs?

- CFGs generate constituent-based representations
  - Noun phrases, verb phrases, etc.
  - These tell us about the **syntactic** structure
- Dependency grammars define sentence structure in terms of the **semantic** relationships between individual words
  - Nominal subject, direct object, etc.
- For both, labels are still drawn from a fixed inventory of grammatical relations

Dependency grammars are especially helpful for interpreting morphologically rich languages with a relatively free word order.

Morphologically rich: Grammatical relationships are indicated by changes to words, rather than sentence position

Free word order: Words can be moved around in a sentence but the overall meaning will remain the same (less reliance on syntax)

Typically, languages that are morphologically richer have less strict syntactic rules

This Week's Topics

Dependency Structure
Transition-Based Dependency Parsing
Graph-Based Dependency Parsing
Meaning Representations

Tuesday

Thursday

Model-Theoretic Semantics
First-Order Logic
Semantic Roles
Semantic Role Labeling
Selectional Preferences

This Week's Topics

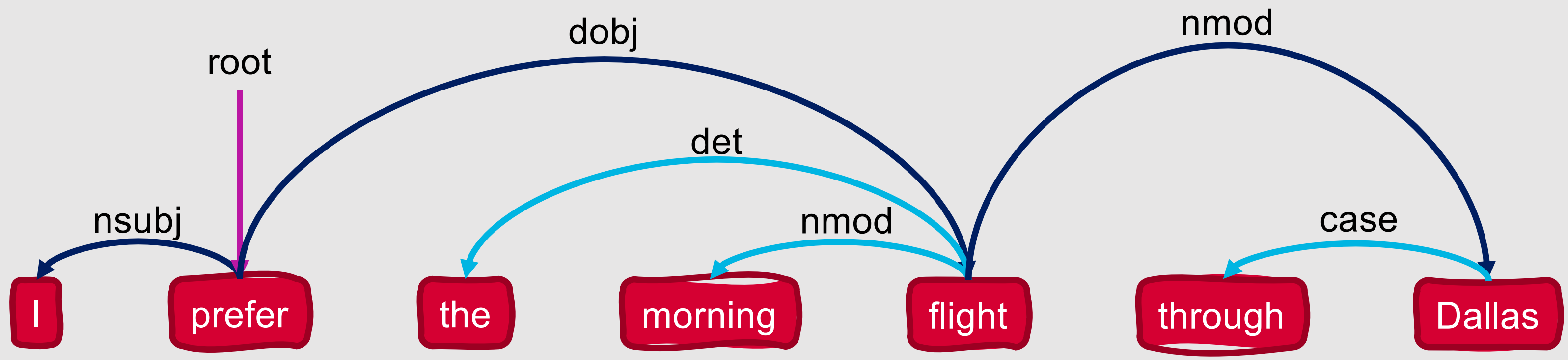
Tuesday

- Dependency Structure
- Transition-Based Dependency Parsing
- Graph-Based Dependency Parsing
- Meaning Representations

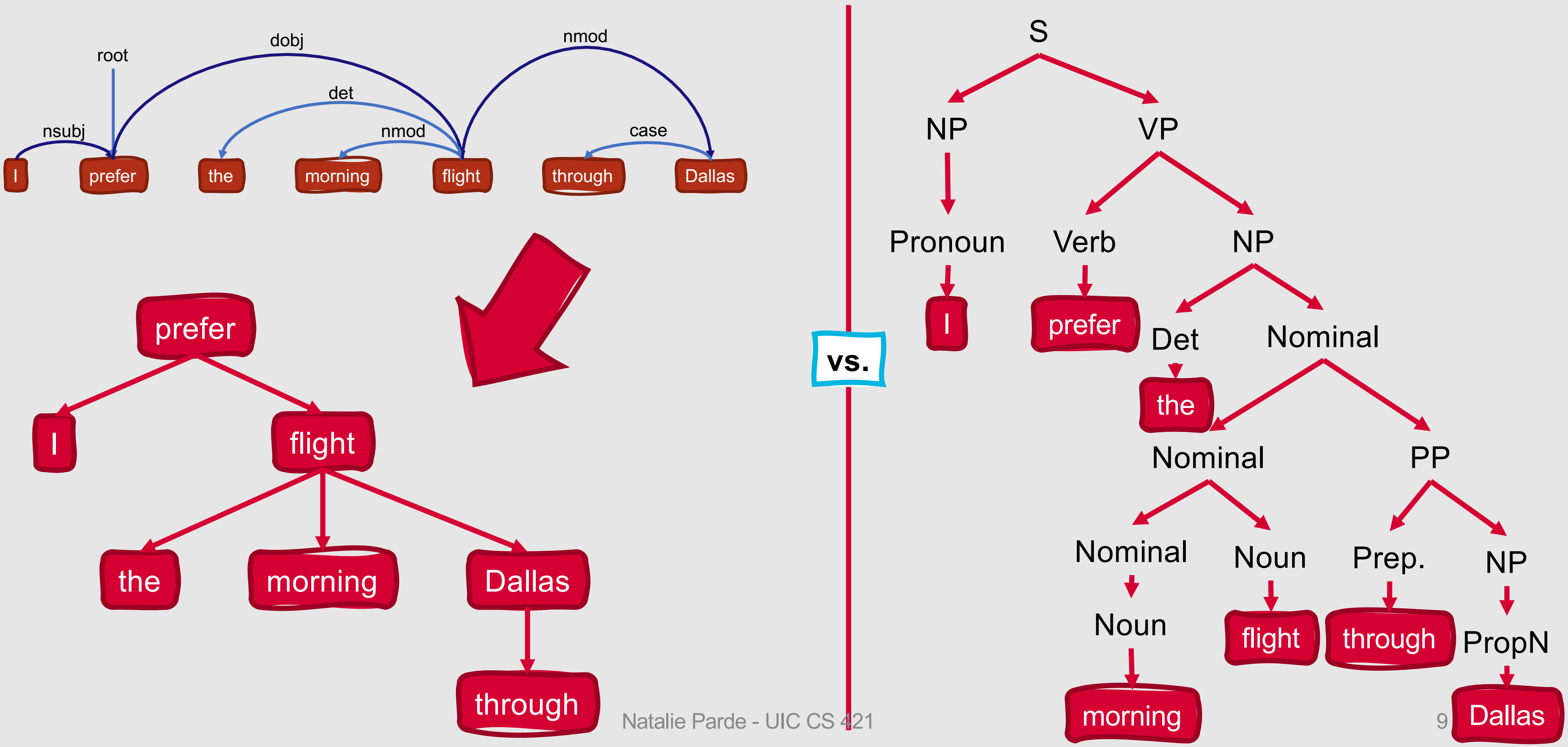
Thursday

- Model-Theoretic Semantics
- First-Order Logic
- Semantic Roles
- Semantic Role Labeling
- Selectional Preferences

Typed Dependency Structure



Comparison with Syntactic Parse



Dependency Relations

- **Heads** are linked to the words that are immediately **dependent** on them
- Relation types describe the **dependent**'s role with respect to its **head**
  - Subject
  - Direct object
  - Indirect object

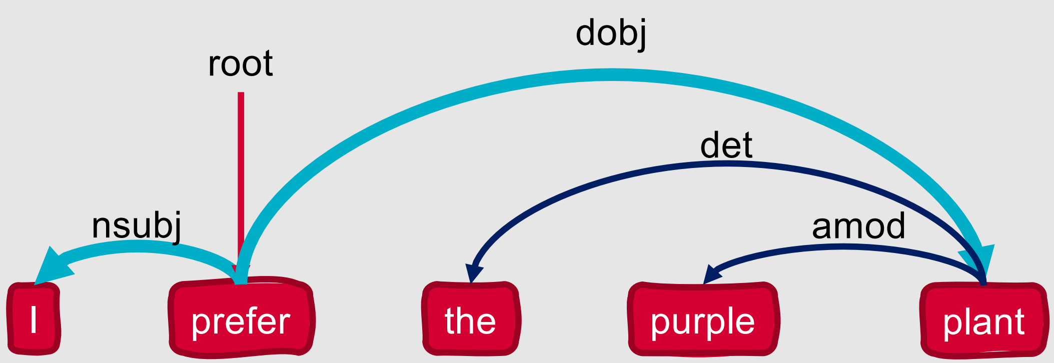
Dependency Relations

- Relation types *tend* to correlate with sentence position and constituent type in English, but there is not an explicit connection between these elements
- In languages with relatively free word order, the information encoded in these relation types often cannot be estimated from constituency trees

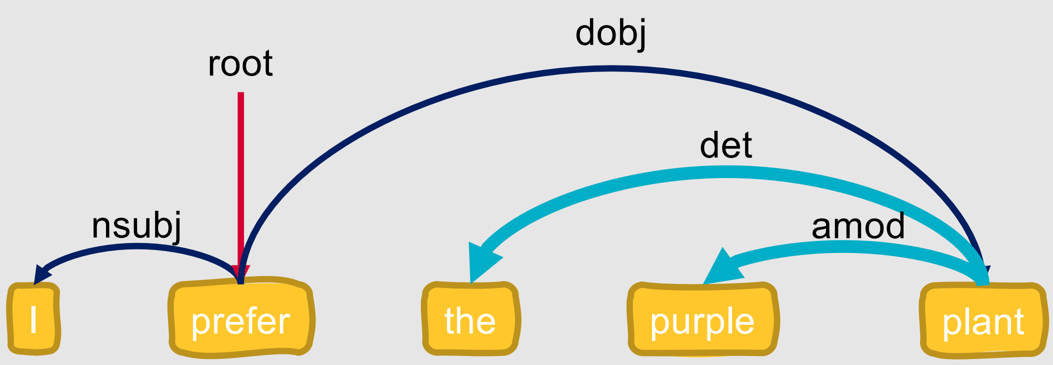
Just like with CFGs, there are a variety of taxonomies that can be used to label dependencies between words.

- A couple of the most popular **dependency treebanks and tagsets** include:
  - Stanford dependencies
    - https://downloads.cs.stanford.edu/nlp/software/dependencies_manual.pdf
  - Universal dependencies
    - https://universaldependencies.org/
    - Most popular tagset recently!
    - Dependencies can be categorized as:
      - **Clausal Relations:** Describe syntactic roles that say something about the predicate
      - **Modifier Relations:** Describe the ways that words can modify their heads

Clausal Relations



Modifier Relations



Universal Dependencies

Structural categories of dependent

Functional categories w.r.t. head

	Nominals	Clauses	Modifier Words	Function Words
Core Arguments of Clausal Predicates	nsubj obj iobj	csubj ccomp xcomp		
Non-Core Dependents of Clausal Predicates	obl vocative expl dislocated	advcl	advmod discourse	aux cop mark
Dependents of Nominals	nmod appos nummod	acl	amod	det clf case

Other miscellaneous dependency relations (see https://universaldependencies.org/u/dep/index.html for details):
conj, cc, fixed, flat, compound, list, parataxis, orphan, goeswith, reparandum, punct, root, dep

Universal Dependencies

Structural categories of dependent

Functional categories w.r.t. head

	Nominals	Clauses	Modifier Words	Function Words
Core Arguments of Clausal Predicates	nsubj obj iobj	xcomp		
Non-Core Dependents of Clausal Predicates	obl vocative expl dislocated		od rse	aux cop mark
Dependents of Nominals	nmod appos nummod			det clf case

Natalie wrote a dissertation.
nsubj(wrote, Natalie)

Natalie wrote a dissertation.
obj(wrote, dissertation)

Natalie wrote UIC a dissertation.
iobj(wrote, UIC)

Other miscellaneous dependency relations (see https://universaldependencies.org/u/dep/index.html for details):
conj, cc, fixed, flat, compound, list, parataxis, orphan, goeswith, reparandum, punct, root, dep

Universal Dependencies

Functional categories w.r.t. head

Structural categories of dependent

	Nominals	Clauses	Modifier Words	Function Words
Core Arguments of Clausal Predicates	nsubj obj iobj	xcomp		
Non-Core Dependents of Clausal Predicates	obl vocative expl dislocated		mod ourse	aux cop mark
Dependents of Nominals	nmod appos nummod			et lf case

Natalie wrote a dissertation for UIC.
obl(wrote, UIC)

UIC, read my dissertation!
vocative(read, UIC)

There is nothing but praise for the dissertation.
expl(nothing, there)

You must not eat it, the dissertation.
dislocated(eat, dissertation)

Other miscellaneous dependency relations (see https://universaldependencies.org/u/dep/index.html for details):
conj, cc, fixed, flat, compound, list, parataxis, orphan, goeswith, reparandum, punct, root, dep

Universal Dependencies

Structural categories of dependent

Functional categories w.r.t. head

	Nominals	Clauses	Modifier Words	Function Words
Core Arguments of Clausal Predicates	nsubj obj iobj			
Non-Core Dependents of Clausal Predicates	obl vocative expl dislocated		e	aux cop mark
Dependents of Nominals	nmod appos nummod		d	det clf case

The purpose of this dissertation is to determine the best homework strategy.
nmod(purpose, dissertation)

My school, UIC, is in Chicago.
appos(school, UIC)

UIC has 34,000 students.
nummod(students, 34,000)

Other miscellaneous dependency relations (see https://universaldependencies.org/u/dep/index.html for details): conj, cc, fixed, flat, compound, list, parataxis, orphan, goeswith, reparandum, punct, root, dep

Universal Dependencies

Structural categories of dependent

Functional categories w.r.t. head

	Nominals	Clauses	Modifier Words	Function Words
Core Arguments of Clausal Predicates	nsubj obj iobj	csubj ccomp xcomp		
Non-Core Dependents of Clausal Predicates	obl vocative expl dislocated	advcl		aux
Dependents of Nominals	nmod appos nummod	acl		det

What she said about starting the project makes sense.
csubj(makes, said)

She said you should start it now.
ccomp(said, start)

I consider it already done.
xcomp(consider, done)

Other miscellaneous dependency relations (see https://universaldependencies.org/u/dep/index.html for details):
conj, cc, fixed, flat, compound, list, parataxis, orphan, goeswith, reparandum, punct, root, dep

Universal Dependencies

Structural categories of dependent

Functional categories w.r.t. head

	Nominals	Clauses	Modifier Words	Function Words
Core Arguments of Clausal Predicates	nsubj obj iobj	csubj ccomp xcomp		
Non-Core Dependents of Clausal Predicates	obl vocative expl dislocated	advcl	advmod discourse	aux cop mark
Dependents of Nominals	nmod appos nummod	acl	amod	det clf case

He was upset when she read her dissertation to him.
advcl(upset, read)

Other miscellaneous dependency relations (see https://universaldependencies.org/u/dep/index.html for details):
conj, cc, fixed, flat, compound, list, parataxis, orphan, goeswith, reparandum, punct, root, dep

Universal Dependencies

Structural categories of dependent

Functional categories w.r.t. head

	Nominals	Clauses	Modifier Words	Function Words
Core Arguments of Clausal Predicates	nsubj obj iobj	csubj ccomp xcomp		
Non-Core Dependents of Clausal Predicates	obl vocative expl dislocated	advcl		mark
Dependents of Nominals	nmod appos nummod	acl	amod	det clf case

There is a document discussing the assignment.
acl(document, discussing)

Other miscellaneous dependency relations (see https://universaldependencies.org/u/dep/index.html for details):
conj, cc, fixed, flat, compound, list, parataxis, orphan, goeswith, reparandum, punct, root, dep

Universal Dependencies

Structural categories of dependent

Functional categories w.r.t. head

	Nominals	Clauses	Modifier Words	Function Words
Core Arguments of Clausal Predicates				
Non-Core Dependents of Clausal Predicates	obl vocative expl dislocated	advcl	advmod discourse	aux cop mark
Dependents of Nominals			amod	det clf case

UIC quickly emailed the students about the day off.
advmod(emailed, quickly)

She said, “Well, let’s schedule a meeting.”
discourse(schedule, well)

Other miscellaneous dependency relations (see https://universaldependencies.org/u/dep/index.html for details):
conj, cc, fixed, flat, compound, list, parataxis, orphan, goeswith, reparandum, punct, root, dep

Universal Dependencies

Structural categories of dependent

Functional categories w.r.t. head

	Nominals	Clauses	Modifier Words	Function Words
Core Arguments of Clausal Predicates	nsubj obj iobj	csubj ccomp xcomp		
Non-Core Dependents of Clausal Predicates	expl dislocated	advcl	advmod discourse	aux cop mark
Dependents of Nominals	nmod appos nummod	acl	amod	det clf case

He read the extensive syllabus.
amod(syllabus, extensive)

Other miscellaneous dependency relations (see https://universaldependencies.org/u/dep/index.html for details): conj, cc, fixed, flat, compound, list, parataxis, orphan, goeswith, reparandum, punct, root, dep

Universal Dependencies

Structural categories of dependent

Functional categories w.r.t. head

	Nominals	Clauses	Modifier Words	Function Words
Core Arguments of Clausal Predicates	nsubj	csubj		
Non-Core Dependents of Clausal Predicates	obl vocative expl dislocated			aux cop mark
Dependents of Nominals	nummod			det clf case

UIC had closed the campus for the break.
aux(closed, had)

It was good to have some time off.
cop(good, was)

They knew that this would refresh everyone for the spring.
mark(refresh, that)

Other miscellaneous dependency relations (see https://universaldependencies.org/u/dep/index.html for details): conj, cc, fixed, flat, compound, list, parataxis, orphan, goeswith, reparandum, punct, root, dep

Universal Dependencies

Structural categories of dependent

Functional categories w.r.t. head

	Nominals	Clauses	Modifier Words	Function Words
Core Arguments of Clausal Predicates	nsubj obj iobj	csubj ccomp xcomp		
Non-Core Dependents of Clausal Predicates	obl vocative dislocated		advmod discourse	aux cop mark
Dependents of Nominals	nmod appos nummod	acl	amod	det clf case

That was the goal.
det(goal, the)

A word that accompanies a noun to reflect some conceptual classification of the noun (not used in English)

Everyone went on vacation after that.
case(that, after)

Other miscellaneous dependency relations (see https://universaldependencies.org/u/dep/index.html for details):
conj, cc, fixed, flat, compound, list, parataxis, orphan, goeswith, reparandum, punct, root, dep

Dependency Formalisms

Dependency structures are directed graphs

- $G = (V, A)$
- Vertices (V) correspond to the words in a sentence
  - May also include punctuation
  - In morphologically rich languages, may include stems and affixes
- Arcs (A) are ordered pairs of vertices that capture the grammatical relationships between those words

In general, dependency structures:

- Must be connected
- Must have a designated root node with no incoming arcs
- Must be acyclic

Additional Notes

- All vertices *except the root node* have exactly one incoming arc
- There is a unique path from the root node to each vertex

This Week's Topics

Tuesday

- Dependency Structure
- Transition-Based Dependency Parsing
- Graph-Based Dependency Parsing
- Meaning Representations

Thursday

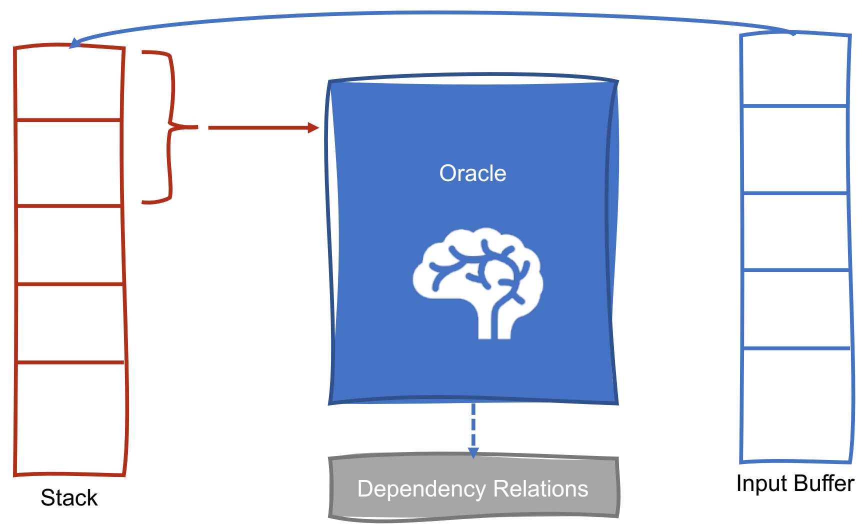
- Model-Theoretic Semantics
- First-Order Logic
- Semantic Roles
- Semantic Role Labeling
- Selectional Preferences

Types of Dependency Parsers

Transition	**Transition-based** • Build a single tree in a beginning-to-end sweep over the input sentence
Graph	**Graph-based** • Search through the space of possible trees for a given sentence, and try to find the tree that maximizes some score

Transition-based Dependency Parsing

- Earliest transition-based approach: **shift-reduce parsing**
  - Input tokens are successively shifted onto a stack
  - The two top elements of the stack are matched against a set of possible relations provided by some knowledge source
  - When a match is found, a head-dependent relation between the matched elements is asserted
- Goal is to find a final parse that accounts for all words



Transition-based Parsing

- We can define **transition operators** to guide the parser's decisions
- Transition operators work by producing new **configurations**:
  - Stack
  - Input buffer of words
  - Set of relations representing a dependency tree

Transition-based Parsing

Initial configuration:

- Stack contains the ROOT node
- Input buffer is initialized with all words in the sentence, in order
- Empty set of relations represents the parse

Final configuration:

- Stack should be empty (except ROOT)
- Input buffer should be empty
- Set of relations represents the parse

Operators

- The operators used in transition-based parsing then perform one of the following tasks:
  - **Assign the current word as the head of some other word** that has already been seen
  - **Assign some other word that has already been seen as the head** of the current word
  - **Do nothing** with the current word

Operators

- More formally, these operators are defined as:
  - **LeftArc:** Asserts a head-dependent relation between the word at the top of the stack and the word directly beneath it (the second word), and removes the second word from the stack
    - Cannot be applied when ROOT is the second element in the stack
    - Requires two elements on the stack
  - **RightArc:** Asserts a head-dependent relation between the second word and the word at the top of the stack, and removes the word at the top of the stack
    - Requires two elements on the stack
  - **Shift:** Removes a word from the front of the input buffer and pushes it onto the stack

Arc Standard Approach to Transition-based Parsing

- These operators implement the **arc standard approach** to transition-based parsing
- Notable characteristics:
  - Transition operators only assert relations between elements at the top of the stack
  - Once an element has been assigned its head, it is removed from the stack
    - Not available for further processing!
  - The arc standard approach is a **greedy algorithm**
- Benefits:
  - Reasonably effective
  - Simple to implement

Formal Algorithm: Arc Standard Approach

```
state ← {[root], [words], []}
while state not final:
     # Choose which transition operator to apply
    transition ← oracle(state)

     # Apply the operator and create a new state
    state ← apply(transition, state)
```

Process ends when:
- All words in the sentence have been consumed
- The ROOT node is the only element remaining on the stack

Arc Standard: Example

Input Buffer	book	me	the	morning	flight	
Stack	root					

Relations

Arc Standard: Example

Input Buffer

	me	the	morning	flight

Stack

book	root				

Relations

Only one item in the stack!

Shift **book** from the input buffer to the stack

Arc Standard: Example

Input Buffer: | | | the | morning | flight |

Stack: | me | book | root | | | |

Relations

Valid options: Shift, RightArc

Oracle selects Shift

Shift **me** from the input buffer to the stack

Arc Standard: Example

Input Buffer			the	morning	flight

Stack	book	root				

Relations

(book → me)

Valid options: Shift, RightArc, LeftArc

Oracle selects RightArc

Remove **me** from the stack

Add relation (book → me) to the set of relations

Arc Standard: Example

Input Buffer

			morning	flight

Stack

the	book	root			

Relations

(book → me)

Valid options: Shift, RightArc

Oracle selects Shift

Shift **the** from the input buffer to the stack

Arc Standard: Example

Input Buffer

				flight

Stack

morning	the	book	root		

Relations

(book → me)

Valid options: Shift, RightArc, LeftArc

Oracle selects Shift

Shift **morning** from the input buffer to the stack

Arc Standard: Example

Input Buffer						
Stack	flight	morning	the	book	root	

Relations: (book → me)

Valid options: Shift, RightArc, LeftArc

Oracle selects Shift

Shift **flight** from the input buffer to the stack

Arc Standard: Example

Input Buffer						
Stack	flight	the	book	root		

Relations

(book → me)
(flight → morning)

Valid options: RightArc, LeftArc

Oracle selects LeftArc

Remove **morning** from the stack

Add relation (flight → morning) to the set of relations

Arc Standard: Example

Input Buffer

Stack

flight	book	root			

Relations

(book → me)
(flight → morning)
(flight → the)

Valid options: RightArc, LeftArc

Oracle selects LeftArc

Remove **the** from the stack

Add relation (flight → the) to the set of relations

Arc Standard: Example

Input Buffer

Stack

book	**root**				

Relations

(book → me)
(flight → morning)
(flight → the)
(book → flight)

Valid options: RightArc, LeftArc

Oracle selects RightArc

Remove **flight** from the stack

Add relation (book → flight) to the set of relations

Arc Standard: Example

Input Buffer

Stack

root

Relations

(book → me)
(flight → morning)
(flight → the)
(book → flight)
(root → book)

Valid options: RightArc

Oracle selects RightArc

Remove **book** from the stack

Add relation (root → book) to the set of relations

Arc Standard: Example

Input Buffer

Stack

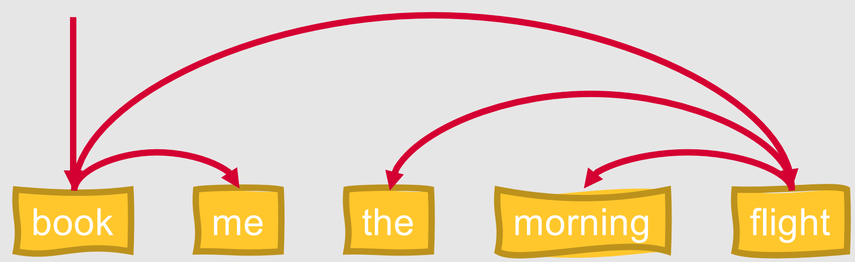
root

Valid options: None

State is final

Relations

(book → me)
(flight → morning)
(flight → the)
(book → flight)
(root → book)



How do we get actual dependency labels?

- Parameterize **LeftArc** and **RightArc**
  - LeftArc(nsubj), RightArc(obj), etc.
- Of course, this makes the oracle's job more difficult (much larger set of operators from which to choose!)
  - Incorrect choices by the oracle lead to incorrect parses since the algorithm cannot perform any backtracking
  - However, alternate sequences may also lead to equally valid parses

iobj(book → me)
compound(flight → morning)
det(flight → the)
obj(book → flight)
root(root → book)

- Generally, systems use **supervised machine learning** for this task
  - Logistic regression
  - Support vector machines
  - Neural networks
- The oracle learns which transitions to predict for new configurations based on extracted features and/or representations for labeled configurations in the training set

How does the oracle know what to choose?

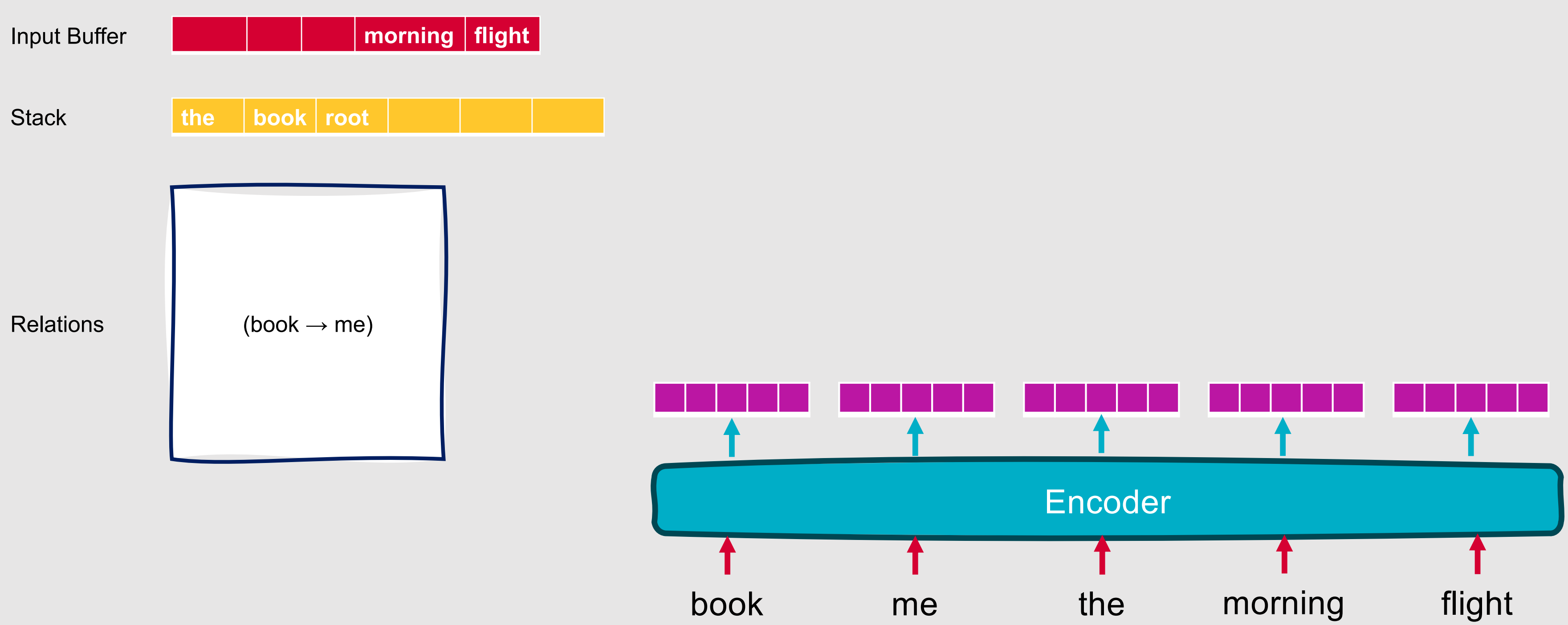
Neural Network-based Oracle

Input Buffer				morning	flight	
Stack	the	book	root			

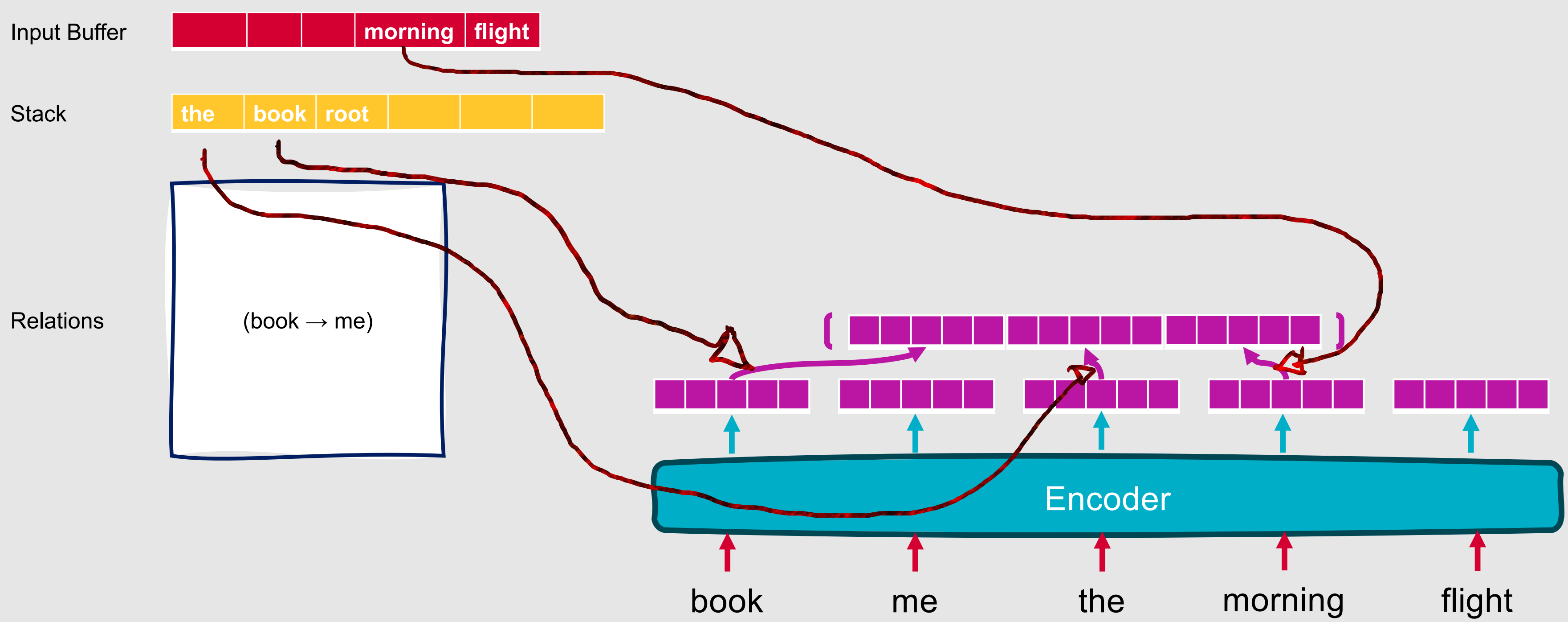
Relations

(book → me)

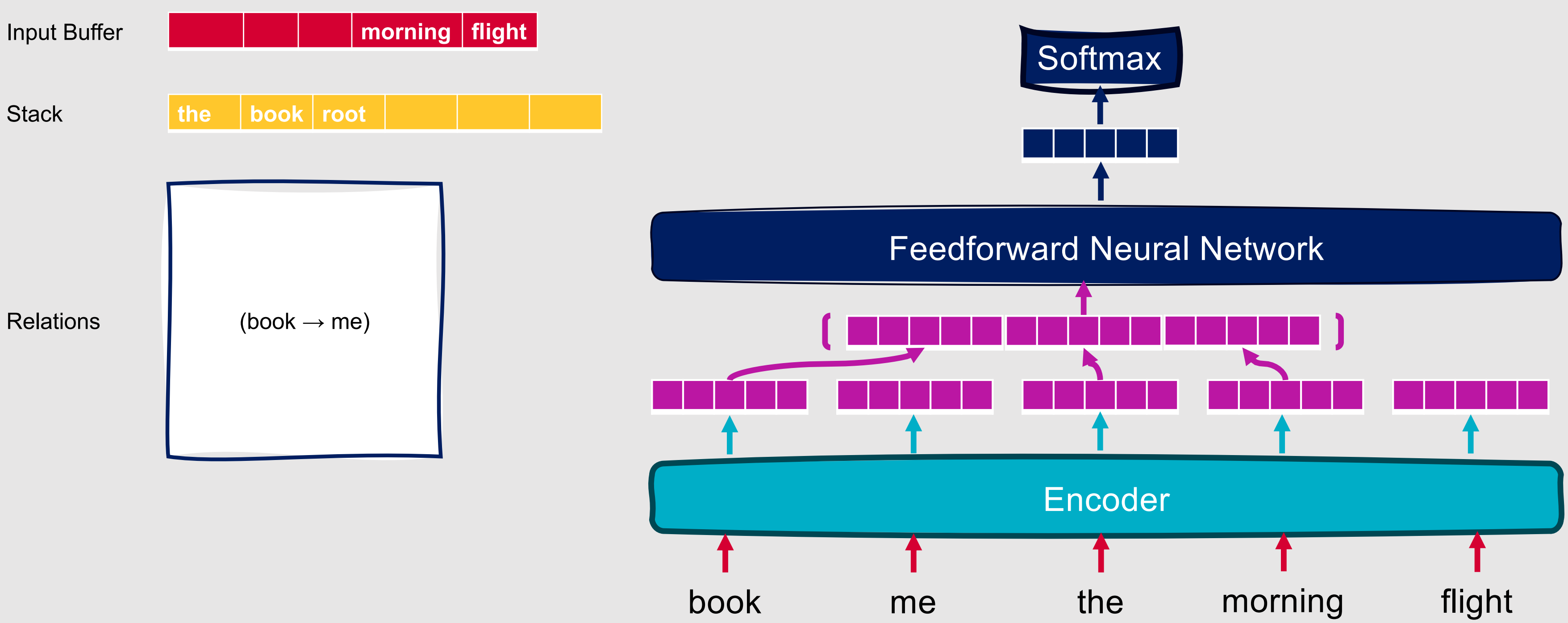
Neural Network-based Oracle



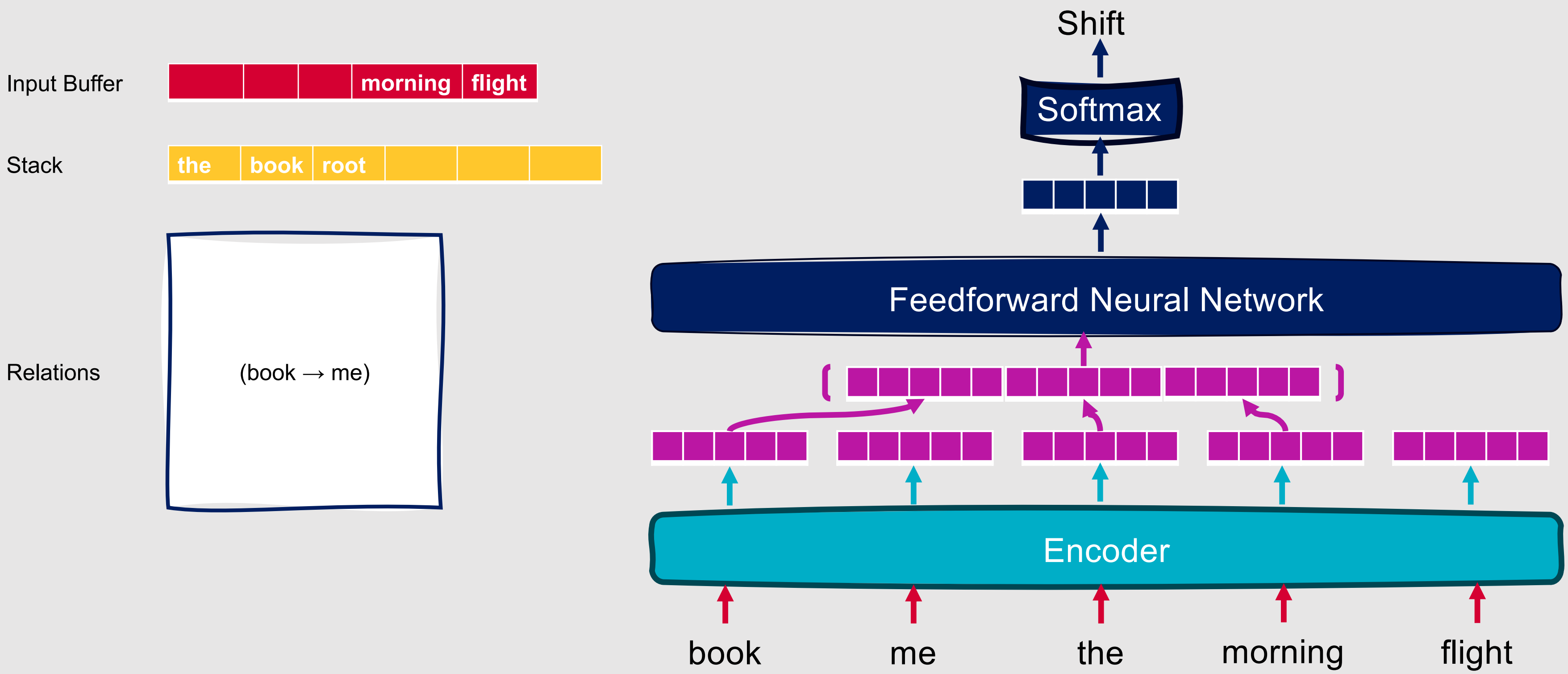
Neural Network-based Oracle



Neural Network-based Oracle



Neural Network-based Oracle



This Week's Topics

Tuesday

- Dependency Structure
- Transition-Based Dependency Parsing
- Graph-Based Dependency Parsing
- Meaning Representations

Thursday

- Model-Theoretic Semantics
- First-Order Logic
- Semantic Roles
- Semantic Role Labeling
- Selectional Preferences

Graph-based Dependency Parsing

- Search through the space of possible dependency trees, attempting to maximize a score based on individual subtrees within the overall tree
- **Edge-factored approaches** determine scores based on the scores of the edges that comprise the tree
  - overall_score(t) = $\sum_{e \in t} score(e)$
  - Letting *t* be a tree for a given sentence, and *e* be its edges

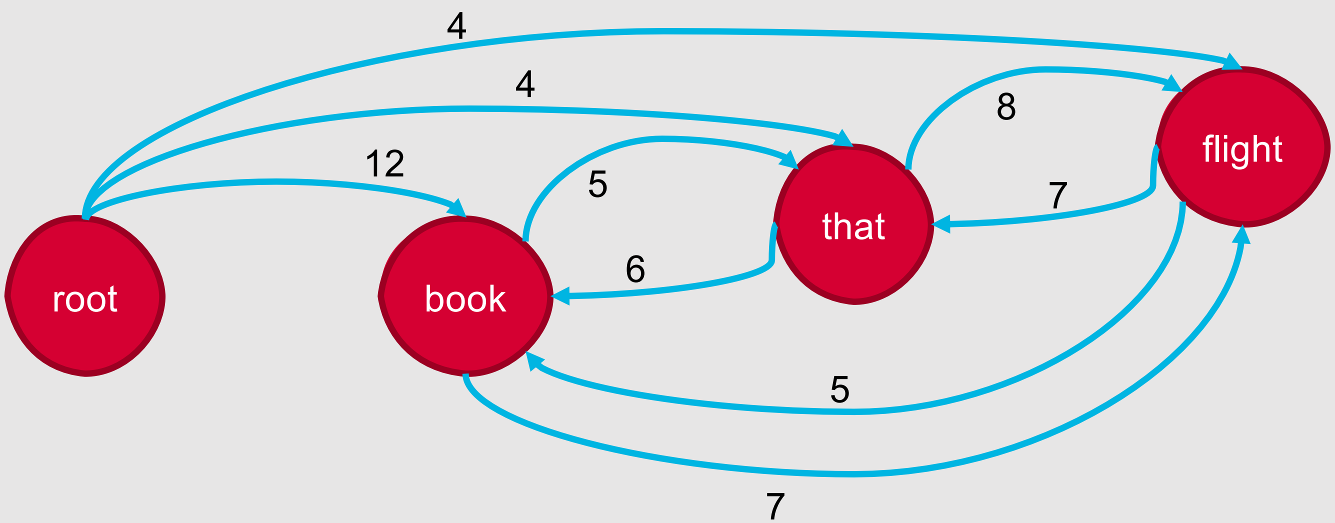
Why use graph-based methods for dependency parsing?

- Since transition-based methods are greedy, they can be fooled by local optima
  - Because of this, they tend to have high accuracy for shorter dependency relations but lower accuracy as the distance between words increases
- Graph-based methods score entire trees, thereby avoiding that issue

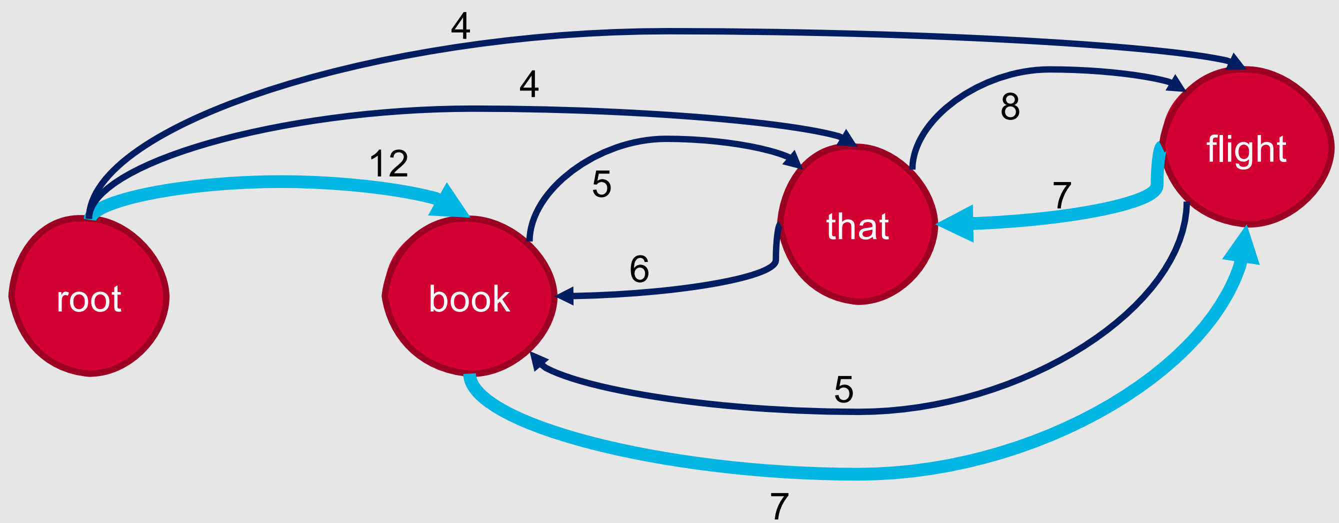
Maximum Spanning Tree

- Given an input sentence, construct a fully-connected, weighted, directed graph
  - Vertices are input words
  - Directed edges represent all possible head-dependent assignments
  - Weights reflect the scores for each possible head-dependent assignment, predicted by a supervised machine learning model
- A maximum spanning tree represents the preferred dependency parse for the sentence, as determined by the weights

Maximum Spanning Tree: Example

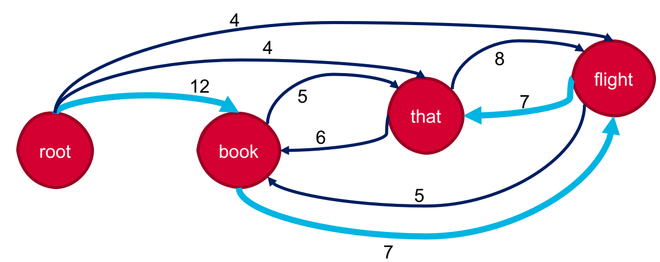


Maximum Spanning Tree: Example



Two things to keep in mind....

- Every vertex in a spanning tree has exactly one incoming edge
- Absolute values of the edge scores are not critical
  - Relative weights of the edges *entering* a vertex are what matter!



How do we know that we have a *maximum* spanning tree?

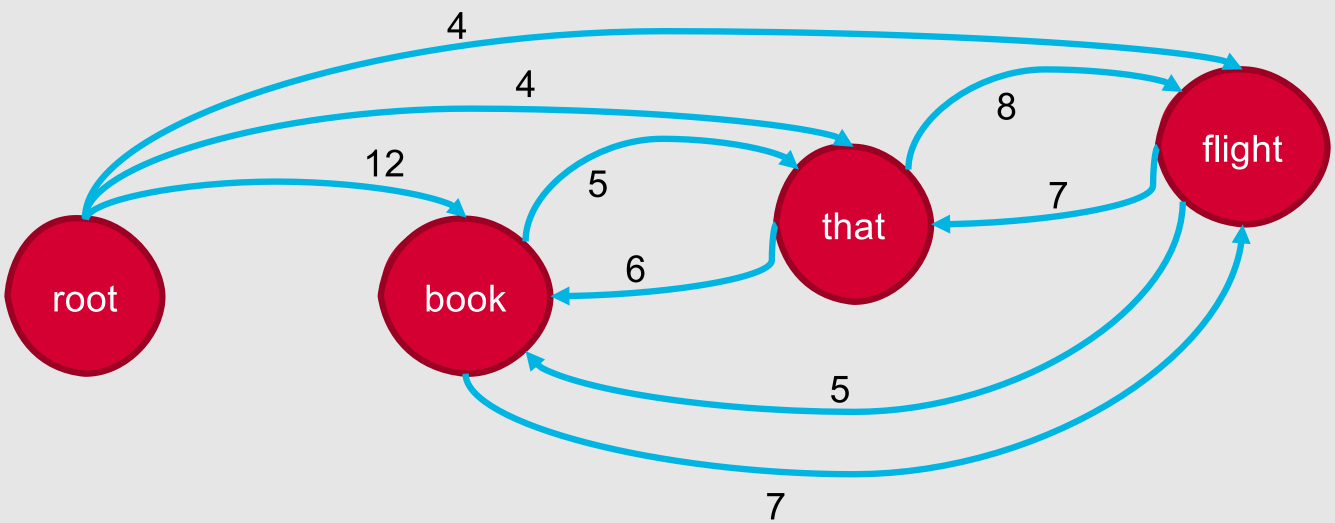
- Given a fully-connected graph G = (V, E), a subgraph T = (V, F) is a spanning tree if:
  - It has no cycles
  - Each vertex (except the root) has exactly one edge entering it
- **If the greedy selection process produces a spanning tree, then that tree is the maximum spanning tree**
- However, the greedy selection process may select edges that result in cycles, which can be addressed by:
  - Collapsing cycles into new nodes, with edges that previously entered or exited the cycle now entering or exiting the new node
  - Recursively applying the greedy selection process to the updated graph until a (maximum) spanning tree is found

Formal Algorithm

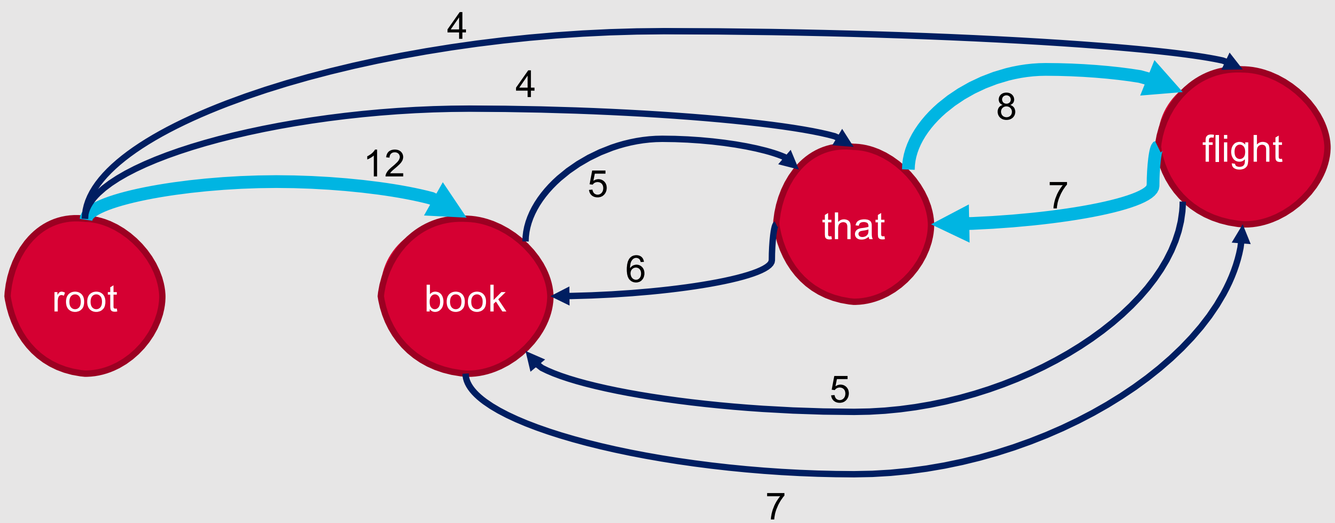
```
F ← []
T ← []
score' ← []
for each v in V do:
        bestInEdge ← argmax_{e=(u,v)∈E} score[e]
        F ← F ∪ bestInEdge
        for each e = (u,v) ∈ E do:
                score'[e] ← score[e] - score[bestInEdge]

        if T=(V,F) is a spanning tree:
                return T
        else:
                C ← a cycle in F
                G' ← collapse(G, C)
                T' ← maxspanningtree(G', root, score') # Recursively call the current function
                T ← expand(T', C)
                return T
```

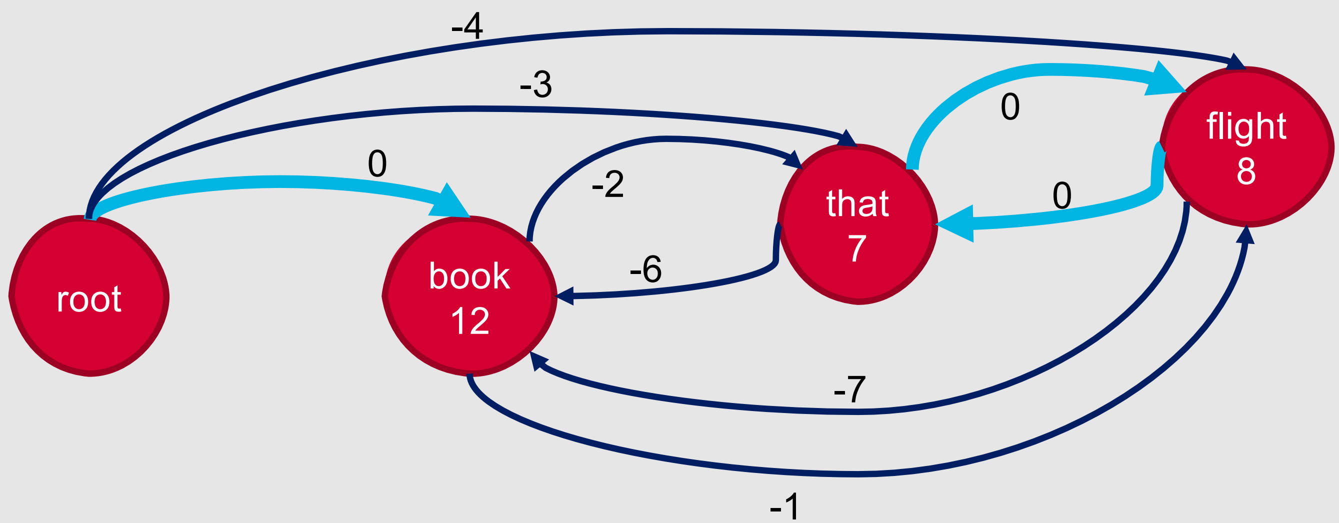
Maximum Spanning Tree: Updated Example



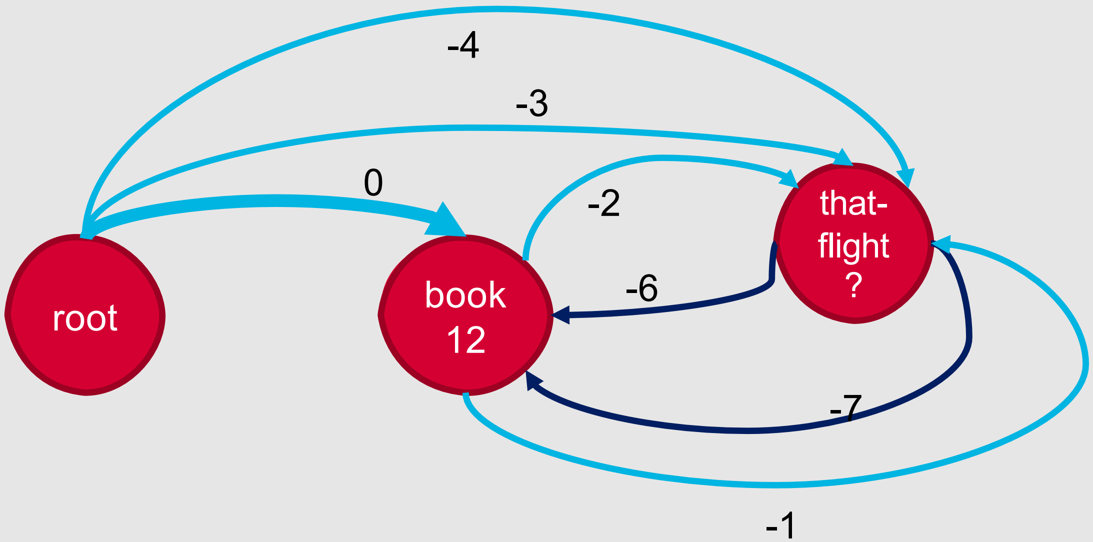
Maximum Spanning Tree: Updated Example



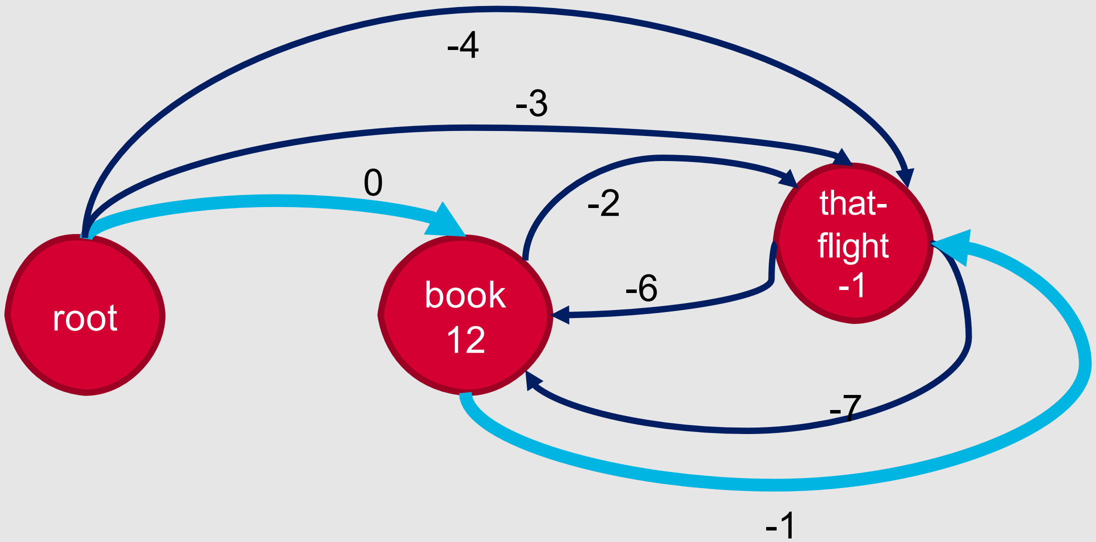
Maximum Spanning Tree: Updated Example



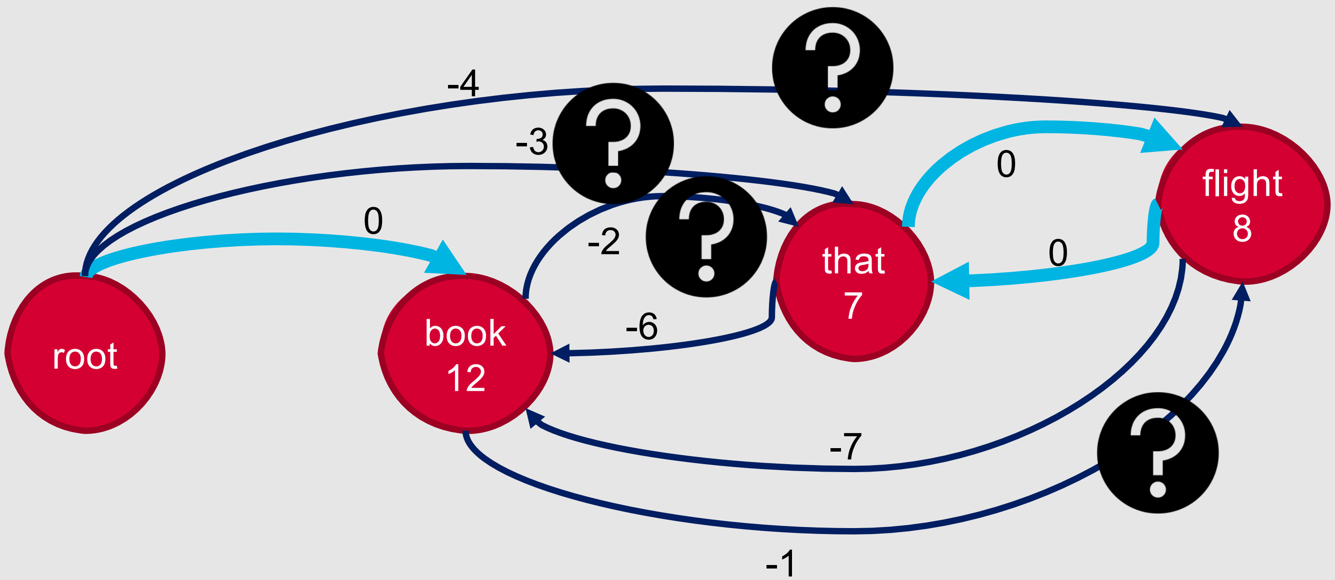
Maximum Spanning Tree: Updated Example



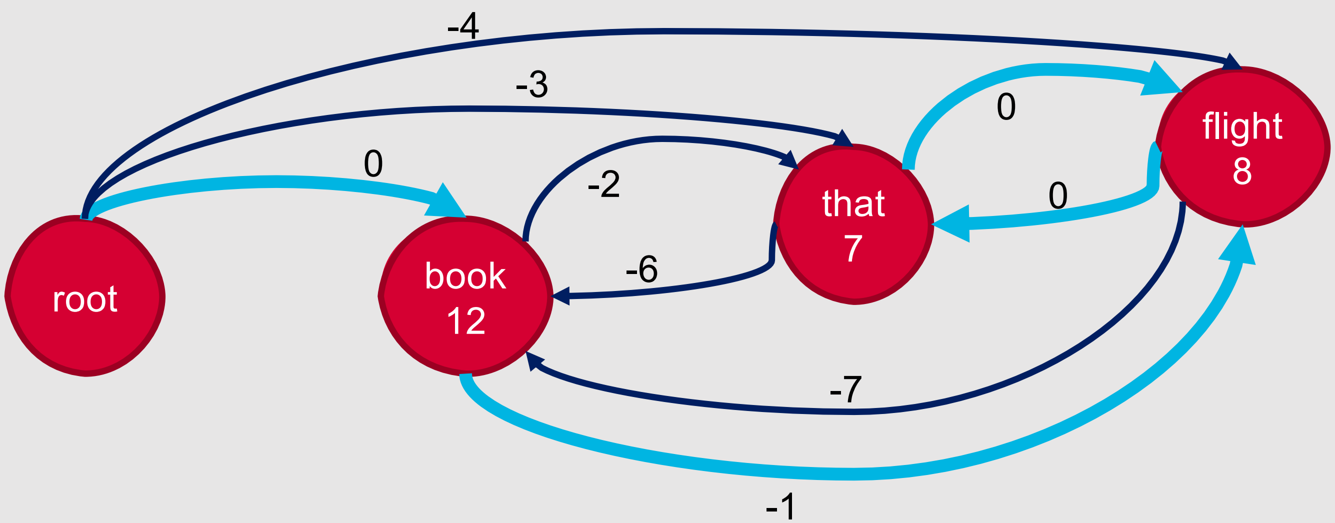
Maximum Spanning Tree: Updated Example



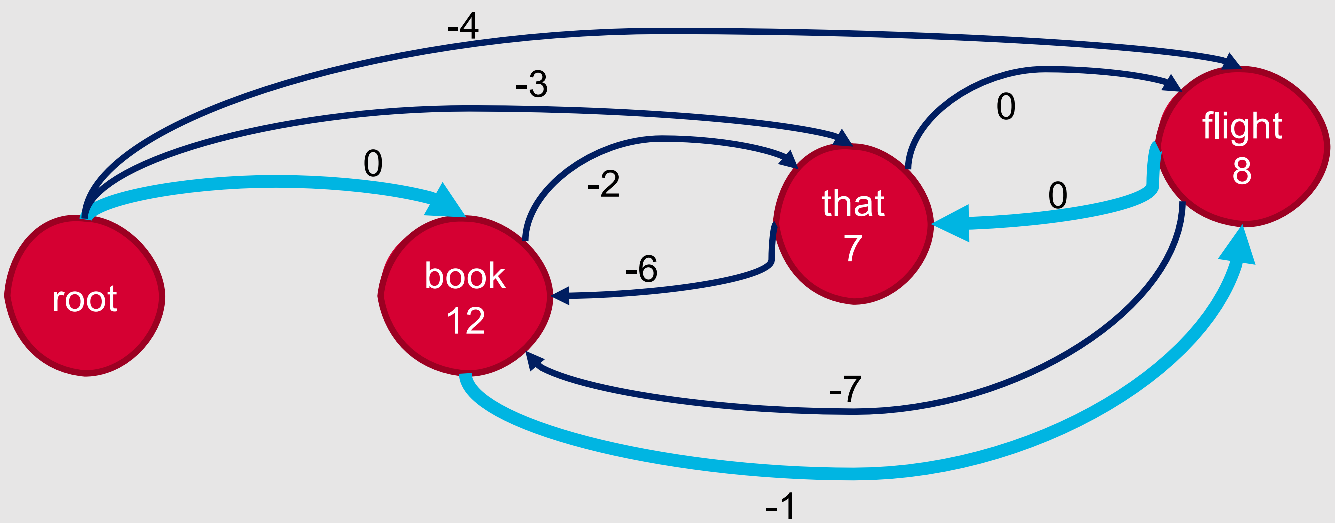
Maximum Spanning Tree: Updated Example



Maximum Spanning Tree: Updated Example



Maximum Spanning Tree: Updated Example



How do we train our model to predict edge weights?

- Similar approach to training the oracle in a transition-based parser
- Feature-based edge scoring models might predict weights based on:
  - Words, lemmas, parts of speech
  - Corresponding features from contexts before and after words
  - Word embeddings
  - Dependency relation type
  - Dependency relation direction
  - Distance from head to dependent
- We can also use neural networks for this process

This Week's Topics

Dependency Structure

Transition-Based Dependency Parsing

Graph-Based Dependency Parsing

Meaning Representations

Tuesday

Thursday

Model-Theoretic Semantics

First-Order Logic

Semantic Roles

Semantic Role Labeling

Selectional Preferences

Why do we need meaning representations?

- Somehow, we need to bridge the gap between **linguistic input** and **world knowledge** to perform semantic processing tasks such as:
  - Answering essay questions on exams
  - Deciding what to order at a restaurant
  - Detecting sarcasm
  - Following recipes
- **Goal: Represent commonsense world knowledge in logical form**

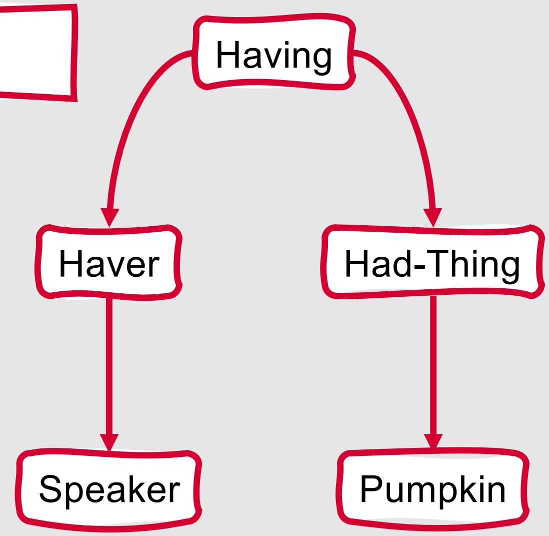
Sample Meaning Representations

I have a pumpkin.

$$\exists x, y \; \text{Having}(x) \wedge \text{Haver}(x, Speaker) \wedge \text{HadThing}(x, y) \wedge \text{Pumpkin}(y)$$

Having

Haver:	Speaker
HadThing:	Pumpkin



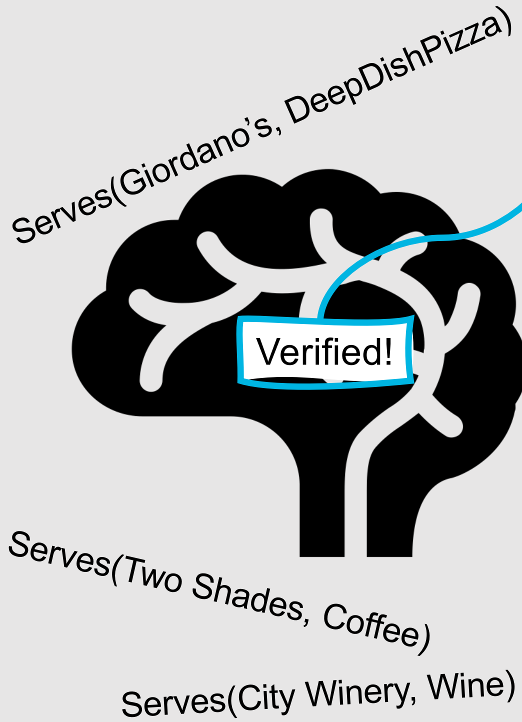
Symbols

- Correspond to **objects**, **properties** of objects, and **relations** among objects
- Symbols link linguistic input (words) to meaning (world knowledge)

Having	
Haver:	Speaker
HadThing:	Pumpkin

Meaning representations should be....

- Verifiable
- Unambiguous
- Able to map to a canonical form
- Supportive of inference and variables
- Expressive



Verifiability

- Computational systems can verify the truth of a meaning representation for a sentence by matching it with **knowledge base** representations
  - **Knowledge Base:** A source of information about the world
- Example proposition: **Giordano's serves deep dish pizza**.
- We can represent this as: **Serves(Giordano's, DeepDishPizza)**
- To verify the truth of this proposition, we would:
  - Search a knowledge base containing facts about restaurants
  - If we found a fact matching this, we have verified the proposition
  - If not, we must assume that the fact is incorrect or, at best, our knowledge base is incomplete

Unambiguous Representations

- Ambiguity does not stop at syntax!
- Semantic ambiguities are everywhere:
  - Sarcasm
  - Idiom
  - Metaphor
  - Hyperbole
- To resolve semantic ambiguities, computational methods must select which from a set of possible interpretations is most correct, given the circumstances surrounding the linguistic input

Let’s devour some building near SEO!

Let’s eat at a restaurant near SEO!

Vagueness

- Closely related to ambiguity
- However, vagueness does not give rise to multiple representations
- In fact, it is advantageous for meaning representations to maintain a certain level of vagueness
  - Otherwise, you may be "overfitting" to your set of example sentences

Canonical Form

- Sentences are ambiguous when they could reasonably be assigned multiple meaning representations
- However, **multiple sentences could also be assigned the same meaning representation**
  - Giordano’s serves deep dish pizza.
  - They have deep dish pizza at Giordano’s.
  - Deep dish pizza is served at Giordano’s.
  - You can eat deep dish pizza at Giordano’s.

Inference and Variables

- It’s impossible for a knowledge base to comprehensively cover all facts about the world, so computational systems also need to be able to draw commonsense inferences based on meaning representations
  - **Will people who like deep dish pizza want to eat at Giordano’s?**
    - We don’t have a fact explicitly specifying that they do, but we can infer that if they like deep dish pizza, they will probably like a restaurant that serves it

Inference and Variables

- **Inference:** A system's ability to draw valid conclusions based on the meaning representations of inputs and its store of background knowledge
- **Variables** allow you to build propositions without requiring a specific instance of something
  - Serves(x, DeepDishPizza)
- These propositions can only be successfully matched by known instances from a knowledge base that would resolve in a truthful entire proposition
  - Serves(x, DeepDishPizza)
    - Serves(Giordano's, DeepDishPizza) 🙂
    - Serves(IDOF, DeepDishPizza) 🤨

Expressiveness

- **Expressive power:** The breadth of ideas that can be represented in a language
- Meaning representations must be **expressive** enough to handle a wide range of subject matter

Summary: Dependency Parsing

- **Dependency parsing** is the process of automatically determining **directed relationships between words** in a source sentence
- Numerous dependency tagsets exist, but currently the most common tagset is the set of **universal dependencies**
- Dependency parsers can be **transition-based** or **graph-based**
- A popular transition-based method is the **arc standard** approach
- A popular graph-based method is the **maximum spanning tree** approach
- Both make use of **supervised machine learning** to aid the decision-making process

This Week's Topics

Dependency Structure
Transition-Based Dependency Parsing
Graph-Based Dependency Parsing
Meaning Representations

Tuesday

Thursday

Model-Theoretic Semantics
First-Order Logic
Semantic Roles
Semantic Role Labeling
Selectional Preferences

Model-Theoretic Semantics

All meaning representation schemes share an ability to represent objects, properties of objects, and relations among objects

A **model** is a formal construct that stands for a particular state of affairs in the world that we're trying to represent

Expressions (words or phrases) in the meaning representation language can be mapped to elements of the model

Relevant Terminology

- Vocabulary
  - **Non-Logical Vocabulary:** Open-ended sets of names for objects, properties, and relations in the world we're representing
  - **Logical Vocabulary:** Closed set of symbols, operators, quantifiers, and links that provide the formal means for composing expressions in the language
- **Domain:** The set of objects that are part of the state of affairs being represented in the model
  - For a given domain, **objects** are elements
    - grapes, violets, plums, CS421, Abari, Meghan
  - **Properties** are sets of elements corresponding to a specific characteristic
    - purple = {grapes, violets, plums}
  - **Relations** are sets of tuples, each of which contain domain elements that take part in a specific relation
    - TAFor = {(CS421, Abari), (CS421, Meghan)}
- **Each object in the non-logical vocabulary corresponds to a unique element in the domain**; however, each element in the domain does not need to be mentioned in a meaning representation

Functions

- We create mappings from non-logical vocabulary to formal denotations using **functions** or interpretations
- Assume that we have:
  - A collection of restaurant patrons and restaurants
  - Various facts regarding the likes and dislikes of patrons
  - Various facts about the restaurants
- In our current state of affairs (our **model**) we're concerned with four patrons designated by the non-logical symbols (**elements**) *Natalie*, *Devika*, *Nikolaos*, and *Mina*
- We'll use the constants *a*, *b*, *c*, and *d* to refer to those respective elements

Example Application

patron = {Natalie, Devika, Nikolaos, Mina} = {a, b, c, d}

- We're also concerned with three restaurants designated by the non-logical symbols *Giordano's*, *IDOF*, and *Artopolis*
- We'll use the constants *e*, *f*, and *g* to refer to those respective elements

Example Application

patron = {Natalie, Devika, Nikolaos, Mina} = {a, b, c, d}

restaurants = {Giordano's, IDOF, Artopolis} = {e, f, g}

- Finally, we'll assume that our model deals with three cuisines in general, designated by the non-logical symbols *Italian*, *Mediterranean*, and *Greek*
- We'll use the constants *i*, *j*, and *k* to refer to those elements

Example Application

patron = {Natalie, Devika, Nikolaos, Mina} = {a, b, c, d}

restaurants = {Giordano's, IDOF, Artopolis} = {e, f, g}

cuisines = {Italian, Mediterranean, Greek} = {i, j, k}

- Now, let's assume we need to represent a few properties of restaurants:
  - *Fast* denotes the subset of restaurants that are known to make food quickly
  - *TableService* denotes the subset of restaurants for which a waiter will come to your table to take your order
- We also need to represent a few relations:
  - *Like* denotes the tuples indicating which restaurants individual patrons like
  - *Serve* denotes the tuples indicating which restaurants serve specific cuisines

Example Application

patron = {Natalie, Devika, Nikolaos, Mina} = {a, b, c, d}

restaurants = {Giordano's, IDOF, Artopolis} = {e, f, g}

cuisines = {Italian, Mediterranean, Greek} = {i, j, k}

Fast = {f}
TableService = {e, g}
Likes = {(a, e), (a, f), (a, g), (b, g), (c, e), (d, f)}
Serve = {(e, i), (f, j), (g, k)}

- This means that we have created the domain D = {a, b, c, d, e, f, g, i, j, k}
- We can evaluate representations like *Natalie likes IDOF* or *Giordano's serves Greek* by mapping the objects in the meaning representations to their corresponding domain elements, and any links to the appropriate relations in the model
  - Natalie likes IDOF → a likes f → Like(a, f) 🙂
  - Giordano's serves Greek → e serves k → Serve(e, k) 🤨

Example Application

patron = {Natalie, Devika, Nikolaos, Mina} = {a, b, c, d}

restaurants = {Giordano's, IDOF, Artopolis} = {e, f, g}

cuisines = {Italian, Mediterranean, Greek} = {i, j, k}

Fast = {f}
TableService = {e, g}
Likes = {(a, e), (a, f), (a, g), (b, g), (c, e), (d, f)}
Serve = {(e, i), (f, j), (g, k)}

- Thus, we're just using sets and operations on sets to ground the expressions in our meaning representations
- What about more complex sentences?
  - Nikolaos likes Giordano's and Devika likes Artopolis.
  - Mina likes fast restaurants.
  - Not everybody likes IDOF.

Example Application

patron = {Natalie, Devika, Nikolaos, Mina} = {a, b, c, d}

restaurants = {Giordano's, IDOF, Artopolis} = {e, f, g}

cuisines = {Italian, Mediterranean, Greek} = {i, j, k}

Fast = {f}
TableService = {e, g}
Likes = {(a, e), (a, f), (a, g), (b, g), (c, e), (d, f)}
Serve = {(e, i), (f, j), (g, k)}

- Plausible meaning representations for the previous examples will not map directly to individual entities, properties, or relations!
- They involve:
  - Conjunctions
  - Equality
  - Variables
  - Negations
- What we need are **truth-conditional semantics**
- This is where **first-order logic** is useful

This Week's Topics

Tuesday

- Dependency Structure
- Transition-Based Dependency Parsing
- Graph-Based Dependency Parsing
- Meaning Representations

Thursday

- Model-Theoretic Semantics
- First-Order Logic
- Semantic Roles
- Semantic Role Labeling
- Selectional Preferences

First-Order Logic

A **meaning representation language** (a way to represent knowledge in a way that is computationally verifiable and supports semantic inference)

Term: First-order logic device for representing objects

Constants	Functions	Variables

Common across all types of terms:

Each one can be thought of as a way of pointing to a specific object

First-Order Logic

- Predicates can be put together using **logical connectives**
  - and $\wedge$
  - or $\vee$
  - implies $\rightarrow$
- They can also be **negated**
  - not $\neg$

- **Constants:** Specific objects in the world being described
  - Conventionally depicted as single capitalized letters (A, B) or words (Natalie, Devika)
  - Refer to exactly one object, although objects can have more than one constant that refers to them
- **Functions:** Concepts that are syntactically equivalent to single-argument predicates
  - Can refer to specific objects without having to associate a named constant with them, e.g., LocationOf(Giordano's)
- **Variables:** Provide the ability to make assertions and draw inferences without having to refer to a specific named object
  - Conventionally depicted as single lowercase letters
- **Predicates:** Symbols that refer to the relations between a fixed number of objects in the domain
  - Can have one or more arguments
    - Serve(Giordano's, Italian)
      - Relates two objects
    - Restaurant(Giordano's)
      - Asserts a property of a single object

Variables and Quantifiers

- Two basic operators in first-order logic are:
  - ∃: The existential quantifier
    - Pronounced “there exists”
  - ∀: The universal quantifier
    - Pronounced “for all”
- These two operators make it possible to represent many more sentences!
  - a restaurant → ∃x Restaurant(x)
  - all restaurants → ∀x Restaurant(x)

We can combine these operators with other basic elements of first-order logic to build logical representations of complex sentences.

- Nikolaos likes Giordano's and Devika likes Artopolis.
  - Like(Nikolaos, Giordano's) ∧ Like(Devika, Artopolis)
- Mina likes fast restaurants.
  - ∀x Fast(x) → Like(Mina, x)
- Not everybody likes IDOF.
  - ∃x Person(x) ∧ ¬Like(x, IDOF)

P	Q	¬P	P∧Q	P∨Q	P→Q
False	False	True	False	False	True
False	True	True	False	True	True
True	False	False	False	True	False
True	True	False	True	True	True

Example: Is the following sentence valid according to our model?

patron = {Natalie, Devika, Nikolaos, Mina} = {a, b, c, d}

restaurants = {Giordano's, IDOF, Artopolis} = {e, f, g}

cuisines = {Italian, Mediterranean, Greek} = {i, j, k}

Fast = {f}
TableService = {e, g}
Likes = {(a, e), (a, f), (a, g), (b, g), (c, e), (d, f)}
Serve = {(e, i), (f, j), (g, k)}

Natalie likes Giordano's and Devika likes Giordano's.

Example: Is the following sentence valid according to our model?

patron = {Natalie, Devika, Nikolaos, Mina} = {a, b, c, d}

restaurants = {Giordano's, IDOF, Artopolis} = {e, f, g}

cuisines = {Italian, Mediterranean, Greek} = {i, j, k}

Fast = {f}
TableService = {e, g}
Likes = {(a, e), (a, f), (a, g), (b, g), (c, e), (d, f)}
Serve = {(e, i), (f, j), (g, k)}

Natalie likes Giordano's and Devika likes Giordano's.

Likes(Natalie, Giordano's) ∧ Likes(Devika, Giordano's)

Example: Is the following sentence valid according to our model?

patron = {Natalie, Devika, Nikolaos, Mina} = {a, b, c, d}

restaurants = {Giordano's, IDOF, Artopolis} = {e, f, g}

cuisines = {Italian, Mediterranean, Greek} = {i, j, k}

Fast = {f}
TableService = {e, g}
Likes = {(a, e), (a, f), (a, g), (b, g), (c, e), (d, f)}
Serve = {(e, i), (f, j), (g, k)}

Natalie likes Giordano's and Devika likes Giordano's.

Likes(Natalie, Giordano's) ∧ Likes(Devika, Giordano's)

Likes(a, e) ∧ Likes(b, e)

Example: Is the following sentence valid according to our model?

patron = {Natalie, Devika, Nikolaos, Mina} = {a, b, c, d}

restaurants = {Giordano's, IDOF, Artopolis} = {e, f, g}

cuisines = {Italian, Mediterranean, Greek} = {i, j, k}

Fast = {f}
TableService = {e, g}
Likes = {(a, e), (a, f), (a, g), (b, g), (c, e), (d, f)}
Serve = {(e, i), (f, j), (g, k)}

Natalie likes Giordano's and Devika likes Giordano's.

Likes(Natalie, Giordano's) ∧ Likes(Devika, Giordano's)

Likes(a, e) ∧ Likes(b, e)

☺

Example: Is the following sentence valid according to our model?

patron = {Natalie, Devika, Nikolaos, Mina} = {a, b, c, d}

restaurants = {Giordano’s, IDOF, Artopolis} = {e, f, g}

cuisines = {Italian, Mediterranean, Greek} = {i, j, k}

Fast = {f}
TableService = {e, g}
Likes = {(a, e), (a, f), (a, g), (b, g), (c, e), (d, f)}
Serve = {(e, i), (f, j), (g, k)}

Natalie likes Giordano’s and Devika likes Giordano’s.

Likes(Natalie, Giordano’s) ∧ Likes(Devika, Giordano’s)

Likes(a, e) ∧ Likes(b, e)

Example: Is the following sentence valid according to our model?

patron = {Natalie, Devika, Nikolaos, Mina} = {a, b, c, d}

restaurants = {Giordano's, IDOF, Artopolis} = {e, f, g}

cuisines = {Italian, Mediterranean, Greek} = {i, j, k}

Fast = {f}
TableService = {e, g}
Likes = {(a, e), (a, f), (a, g), (b, g), (c, e), (d, f)}
Serve = {(e, i), (f, j), (g, k)}

Natalie likes Giordano's and Devika likes Giordano's.

Likes(Natalie, Giordano's) ∧ Likes(Devika, Giordano's)

Likes(a, e) ∧ Likes(b, e)

False …not valid!

A few additional notes….

- Formulas involving ∃ are true if there is *any* substitution of terms for variables that results in a formula that is true according to the model
- Formulas involving ∀ are true only if *all* substitutions of terms for variables result in formulas that are true according to the model
- **Modus ponens:** If a conditional statement is accepted (if p then q), and the **antecedent** (p) holds, then the **consequent** (q) may be inferred
  - More formally:

$$\frac{\begin{array}{l}\alpha \\ \alpha \Rightarrow \beta\end{array}}{\beta}$$

Example: Inference

$$\frac{\begin{array}{l}\text{GreekRestaurant}(Artopolis) \\ \forall x \ \text{GreekRestaurant}(x) \Rightarrow \text{Serves}(x, GreekFood)\end{array}}{\text{Serves}(Artopolis, GreekFood)}$$

conditional statement accepted ✓

Example: Inference

antecedent holds (our model says that Artopolis is a Greek restaurant) ✓

$$\frac{\begin{array}{l}\text{GreekRestaurant}(Artopolis) \\ \forall x\ \text{GreekRestaurant}(x) \Rightarrow \text{Serves}(x, GreekFood)\end{array}}{\text{Serves}(Artopolis, GreekFood)}$$

conditional statement accepted ✓

Example: Inference

antecedent holds (our model says that Artopolis is a Greek restaurant) ✓

$$\frac{\text{GreekRestaurant}(Artopolis) \\ \forall x\ \text{GreekRestaurant}(x) \Rightarrow \text{Serves}(x, GreekFood)}{\text{Serves}(Artopolis, GreekFood)}$$

conditional statement accepted ✓

consequent may be inferred 🙂

Events can be particularly challenging to represent in formal logic!

- You may need to:
  - Determine the correct number of roles for the event
  - Represent facts about different roles associated with the event
  - Ensure that all correct (and only correct) inferences can be derived directly from the event representation
- Some events may theoretically take a variable number of arguments
  - Natalie drinks.
  - Natalie drinks tea.
- However, predicates in first-order logic have fixed **arity** (they accept a fixed number of arguments)
  - Can be solved by creating different versions of the same predicate, developing meaning postulates, or allowing "missing" arguments (e.g., "$\exists x$ Drink(Natalie, x)")

States: Conditions or properties that remain unchanged over some period of time

Events: Indicate changes in some state of affairs

Instead of regular variables, we can add event variables.

- **Event variable:** An argument to the event representation that allows for additional assertions to be included if needed
  - $\exists e$ Drink(Natalie, e)
- If we determine that the actor must drink something specific: $\exists e$ Drink(Natalie, e) $\wedge$ Beverage(e, tea)
- More generally, we could define the representation:
  - $\exists e$ Drink(e) $\wedge$ Drinker(e, Natalie) $\wedge$ Beverage(e, tea)
- With this change, there is no need to specify a fixed number of arguments for a given surface predicate

Description Logics

- How to add increased structure to semantics defined by models?
  - **Description Logics:** Different logical approaches that correspond to subsets of first-order logic
- More specific constraints make it possible to model more specific *forms* of inference
- Represent knowledge about:
  - Categories
  - Individuals who belong to those categories
  - Relationships that can hold among those individuals
- **Terminology:** The set of categories comprising a given application domain
- **Ontology:** Hierarchical representation of subset/superset relations among categories

Representation

Restaurant(x) = Restaurant

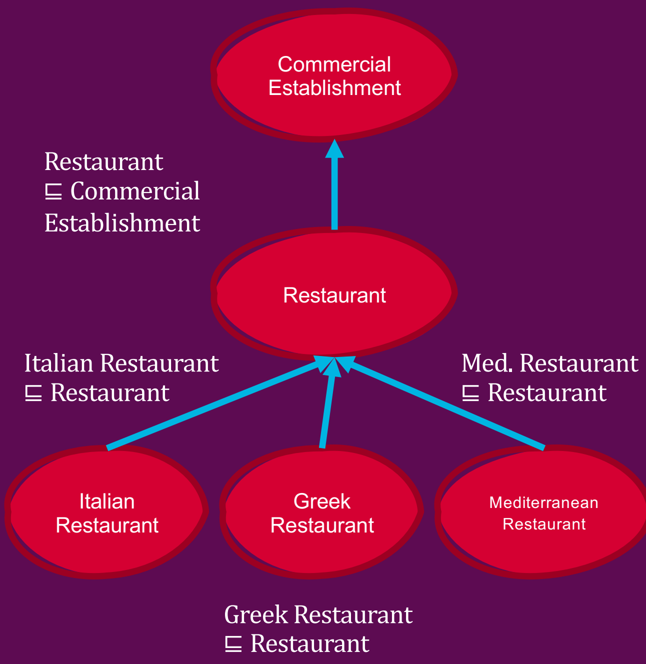
Restaurant(Giordano's) = Restaurant(Giordano's)

First-order logic

Description logics

Hierarchical Structure

- Can be directly specified using subsumption relations between concepts
  - **Subsumption:** All members of category *C* are also members of category *D*, or $C \sqsubseteq D$
- Relations allow us to explicitly define necessary and sufficient conditions for categories
  - $\text{Italian Restaurant} \sqsubseteq \text{Restaurant} \sqcap \exists \text{hasCuisine.ItalianCuisine}$
  - $\text{Greek Restaurant} \sqsubseteq \text{Restaurant} \sqcap \exists \text{hasCuisine.GreekCuisine}$

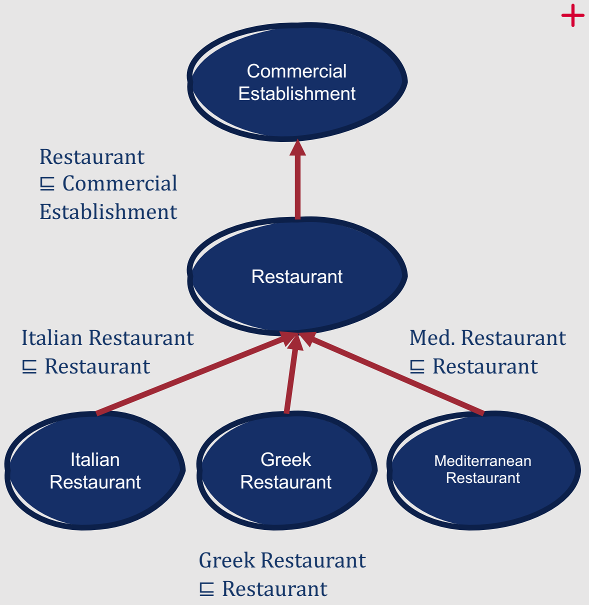


Category Membership

- Coverage or disjointness can be further specified using logical operators
  - Italian Restaurant $\sqsubseteq$ **NOT** Greek Restaurant
  - Restaurant $\sqsubseteq$ **OR** (Italian Restaurant, Greek Restaurant, Mediterranean Restaurant)
- Relations provide further information about category membership
  - Italian Cuisine $\sqsubseteq$ Cuisine
  - Italian Restaurant $\sqsubseteq$ Restaurant $\sqcap$ $\exists$hasCuisine.ItalianCuisine $=$ $\forall x \mathrm{ItalianRestaurant}(x) \longrightarrow \mathrm{Restaurant}(x) \wedge (\exists y \mathrm{Serves}(x, y) \wedge \mathrm{ItalianCuisine}(y))$

Inference

- Subsumption as a form of inference
  - Based on the facts in our terminology, does a superset/subset relationship exist between two concepts?



Real-World Example of Description Logics

- **Web Ontology Language (OWL)**
  - Formally specifies semantic categories of the internet through the creation and deployment of ontologies for application areas of interest
  - Built using a description logic similar to that described in the previous slides

This Week's Topics

Dependency Structure
Transition-Based Dependency Parsing
Graph-Based Dependency Parsing
Meaning Representations

Thursday

Tuesday

Model-Theoretic Semantics
First-Order Logic
Semantic Roles
Semantic Role Labeling
Selectional Preferences

Semantic Roles

- When extracting information from text, it is useful to understand **semantic roles**, or how participants relate to events
  - Who did what?
  - When?
  - Where?
- There are many possible semantic roles, and they are often application- or domain-specific

Recall the meaning representations we've already seen….

Natalie baked the cake.

$\exists e, x, y\ \text{Baking}(e) \wedge \text{Baker}(e, \text{Natalie}) \wedge \text{BakedThing}(e, y) \wedge \text{Cake}(y)$

Recall the meaning representations we've already seen….

Natalie baked the cake.

- Subject of “bake”
- Deep role specific to the “baking” event

$$\exists e, x, y \ \mathrm{Baking}(e) \wedge \mathrm{Baker}(e, \mathrm{Natalie}) \wedge \mathrm{BakedThing}(e, y) \wedge \mathrm{Cake}(y)$$

What if we consider another sentence?

Natalie ate the cake.

- Subject of “ate”
- Deep role specific to the “eating” event

$\exists e, x, y \; \mathrm{Eating}(e) \wedge \mathrm{Eater}(e, \mathrm{Natalie}) \wedge \mathrm{EatenThing}(e, y) \wedge \mathrm{Cake}(y)$

There are commonalities between these roles!

- "Bakers" and "Eaters" are both:
  - Volitional actors
  - Generally animate
  - Have causal responsibility for their events
- Semantic roles (sometimes referred to as **thematic roles**) are how we capture these commonalities more formally

$$\exists e, x, y \ \mathrm{Baking}(e) \land \mathrm{Baker}(e, \mathrm{Natalie}) \land \mathrm{BakedThing}(e, y) \land \mathrm{Cake}(y)$$

Theme

Agent

$$\exists e, x, y \ \mathrm{Eating}(e) \land \mathrm{Eater}(e, \mathrm{Natalie}) \land \mathrm{EatenThing}(e, y) \land \mathrm{Cake}(y)$$

Semantic roles are ancient!

- First formalized by Pāṇini sometime between 700-400 BCE
- More recently formalized in the 1960s
  - Fillmore (1968): https://files.eric.ed.gov/fulltext/ED019631.pdf
  - Gruber (1965): http://www.ai.mit.edu/projects/dm/theses/gruber65.pdf
- No universally agreed-upon roles, but some are common across numerous papers

Common Semantic Roles

THEMATIC ROLE	DEFINITION	EXAMPLE
Agent	The volitional causer of an event	The **waiter** spilled the soup.
Experiencer	The experiencer of an event	**John** has a headache.
Force	The non-volitional causer of the event	The **wind** blows debris from the mall into our yards.
Theme	The participant most directly affected by an event	Only after Benjamin Franklin broke the **ice**….
Result	The end product of an event	The city built a **regulation-size baseball diamond**….
Content	The proposition or content of a propositional event	Mona asked, **"You met Mary Ann at the supermarket?"**
Instrument	An instrument used in an event	He poached catfish, stunning them with a **shocking device**….
Beneficiary	The beneficiary of an event	Whenever Ann Callahan makes hotel reservations for her **boss**….
Source	The origin of the object of a transfer event	I flew in from **Boston**.
Goal	The destination of an object of a transfer event	I drove to **Portland**.

Some sets of semantic roles are finer-grained, whereas others are broader and more abstract

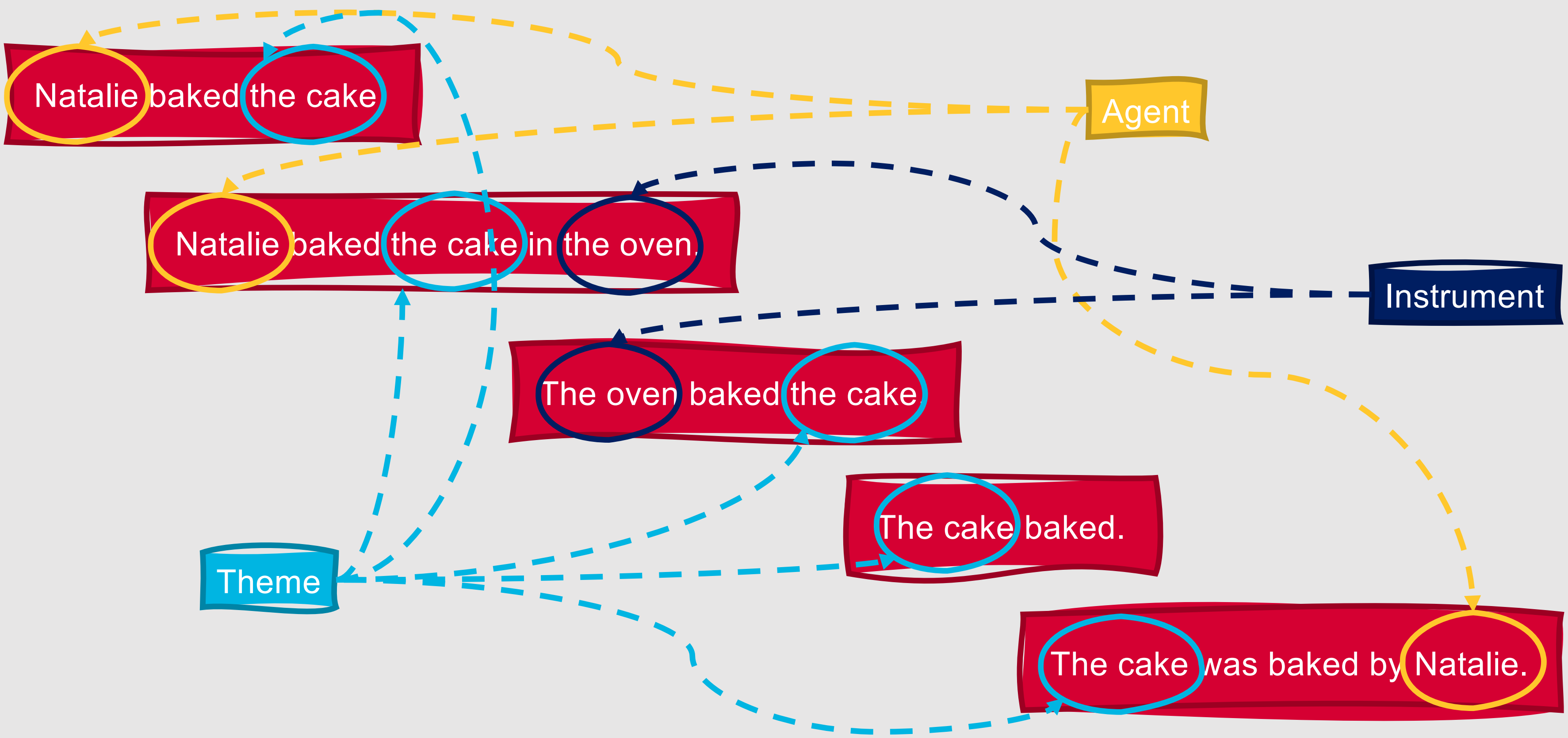
Semantic roles offer another way for us to construct shallow meaning representations.

They allow us to:

- Make inferences that aren't possible from surface representations or parse trees
- Create intermediate languages for downstream tasks

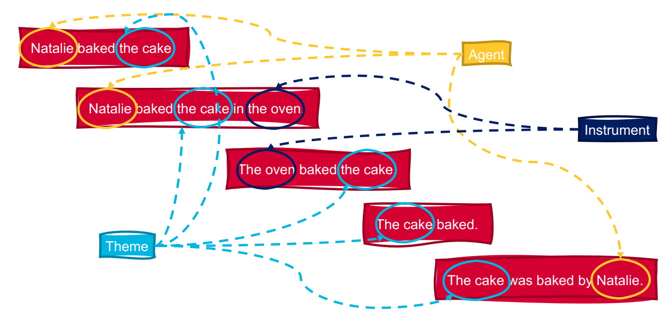
In general, semantic roles help us generalize over different surface realizations of the same predicate arguments

For example….


Natalie baked the cake
Agent
Natalie baked the cake in the oven.
Instrument
The oven baked the cake.
The cake baked.
Theme
The cake was baked by Natalie.

Thematic Grid

- The set of semantic role arguments taken by a verb
  - Also sometimes referred to as a **case frame**
- Semantic roles can often be realized in different syntactic positions
  - For example:
    - Agent=Subject; Theme=Object
    - Instrument=Subject; Theme=Object
    - Theme=Subject
- **Diathesis Alternations:** Alternate acceptable structural realizations for arguments, facilitating generalization over different surface realizations
  - Different verbs can participate in different alternations



Defining Role Sets

- Researchers often find it necessary to fragment more general roles (e.g., Agent) into more specific roles

Instrument

Intermediary: Can appear as subjects

Enabling: Cannot appear as subjects

Conformity to Predefined Properties

- Individual noun phrases may not conform to all properties of an *Agent*, but they might conform to most …can they still be labeled with this role?
  - Might require even more fragmentation!

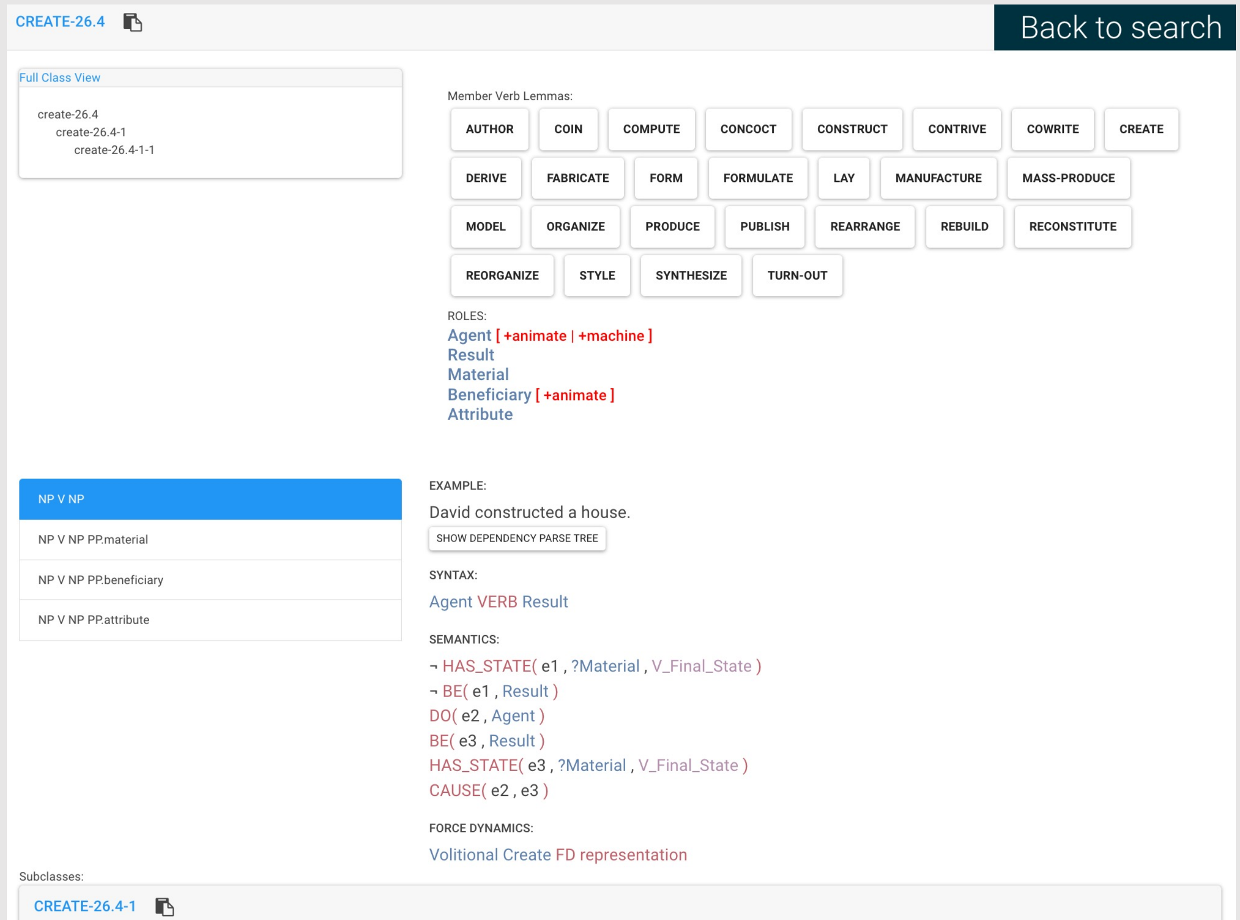
How can these challenges be addressed?

- **Generalized semantic roles**
  - Proto-Agents
  - Proto-Patients
  - Fewer, more abstract roles
- **Semantic roles tailored to specific semantic classes**
  - Additional, more specific roles

VerbNet

- An online resource indicating the semantic classes to which many English verbs belong
- Linked to WordNet and FrameNet entries
- Link: https://verbs.colorado.edu/verbnet/
  - Also an API: https://github.com/cu-clear/verbnet/
  - Also accessible via NLTK: https://www.nltk.org/_modules/nltk/corpus/reader/verbnet.html

VerbNet



CREATE-26.4
Back to search
Full Class View
create-26.4
create-26.4-1
create-26.4-1-1
Member Verb Lemmas:
AUTHOR COIN COMPUTE CONCOCT CONSTRUCT CONTRIVE COWRITE CREATE
DERIVE FABRICATE FORM FORMULATE LAY MANUFACTURE MASS-PRODUCE
MODEL ORGANIZE PRODUCE PUBLISH REARRANGE REBUILD RECONSTITUTE
REORGANIZE STYLE SYNTHESIZE TURN-OUT
ROLES:
Agent [+animate | +machine]
Result
Material
Beneficiary [+animate]
Attribute
NP V NP
NP V NP PP.material
NP V NP PP.beneficiary
NP V NP PP.attribute
EXAMPLE:
David constructed a house.
SHOW DEPENDENCY PARSE TREE
SYNTAX:
Agent VERB Result
SEMANTICS:
¬ HAS_STATE(e1 , ?Material , V_Final_State)
¬ BE(e1 , Result)
DO(e2 , Agent)
BE(e3 , Result)
HAS_STATE(e3 , ?Material , V_Final_State)
CAUSE(e2 , e3)
FORCE DYNAMICS:
Volitional Create FD representation
Subclasses:
CREATE-26.4-1

Semantic Roles

Generalized Semantic Roles

- Abstract over specific thematic roles
- Roles are defined by heuristic features that accompany properties likely to correspond with the generalized class
  - Proto-Agent: Agent-like properties
- More overlapping properties → argument likelier to be labeled with that role

Specialized Semantic Roles

- Define roles that are specific to a particular verb or a group of semantically related verbs or nouns
  - A **Cook** creates a **Produced_food** from (raw) **Ingredients**.
  - The **Heating_instrument** and/or the **Container** may also be specified.

What are some popular resources for semantic role labeling?

PropBank

- https://propbank.github.io/
- Both generalized and verb-specific roles

FrameNet

- https://framenet.icsi.berkeley.edu/fndrupal/
- Semantic roles that are specific to general ideas or *frames*

PropBank

- Proposition Bank
- Available in numerous languages
  - English
  - Hindi
  - Chinese
  - Arabic
  - Finnish
  - Portuguese
  - Basque
  - Turkish

PropBank

- Provides semantic roles associated with different verb senses
- Senses are given numbered arguments as roles
  - Arg0
  - Arg1
  - …
  - ArgN
- PropBank entries:
  - Referred to as **frame files**
  - Definitions for each role are informal glosses

agree.01
- Arg0: Agreer
- Arg1: Proposition
- Arg2: Other entity agreeing

- Ex1: [$_{Arg0}$ The group] agreed [$_{Arg1}$ it wouldn't make an offer].
- Ex2: [$_{ArgM\text{-}TMP}$ Usually] [$_{Arg0}$ John] agrees [$_{Arg2}$ with Mary] [$_{Arg1}$ on everything].

fall.01
- Arg1: Logical subject, patient, thing falling
- Arg2: Extent, amount fallen
- Arg3: start point
- Arg4: end point, end state of arg1

- Ex1: [Arg1 Sales] fell [Arg4 to $25 million] [Arg3 from $27 million].
- [Arg1 The average junk bond] fell [Arg2 by 4.2%].

PropBank can be useful for….

- Recovering shallow semantic information
  - Inferring commonality in event structures for varying surface forms
- Representing modification or adjunct meanings
  - Denoted using non-numbered arguments called **ArgMs**
  - ArgMs aren’t listed in individual frame files since they’re generalizable across predicates

Common Modifier Arguments

ArgM	Description	Example
TMP	When?	Yesterday evening, now
LOC	Where?	At the museum, in Chicago
DIR	Where to/from?	Down, to Chicago
MNR	How?	Clearly, with much enthusiasm
PRP/CAU	Why?	Because, in response to the ruling

PropBank

forecast

forecast.01 - *tell the future*

FORECAST-V NOTES: In the latter example there really should be a trace in objectposition, but treebank didn't put it there. (from forecast.01-v)
FORECAST-N NOTES: Based on sentences in nouns-9998. Comparison to forecast.01-v. No VN class. Framed by Katie. (from forecast.01-n)
FORECASTING-N NOTES: Based on sentences in nouns-9998. Comparison to forecast.01-v. No VN class. Framed by Katie. (from forecasting.01-n)

Aliases:

- forecast (v.)
- forecasting (n.)
- forecast (n.)

Roles:

- ARG0-PAG: *fortune teller*
- ARG1-PPT: *prediction*
- ARG2-PRD: *secondary predication*

transitive

The company forecast that fourth - quarter income from continuing operations would be `` significantly '' lower than a year earlier .

missing object

Saab 's problems were underscored Friday when the company announced that its car division had a 1.2 billion kronor ($ 186.1 million) loss during the first eight months of this year , slightly worse than Saab - Scania had forecast in its first - half report last month .

args 0 and 1

its forecast for economic growth in the EC in 1989

Check out PropBank!

- Link:
  - https://propbank.github.io/
- Paper:
  - Paul Kingsbury and Martha Palmer. From Treebank to PropBank. 2002. In Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC-2002), Las Palmas, Spain.
- PropBank is focused on verbs, but a related project also annotates nominal predicates with the same types of semantic roles:
  - NomBank: https://nlp.cs.nyu.edu/meyers/NomBank.html

Making inferences about semantic commonalities is useful….

- Even more useful: Making inferences across different verbs, or between verbs and nouns
- Potentially applicable to more situations

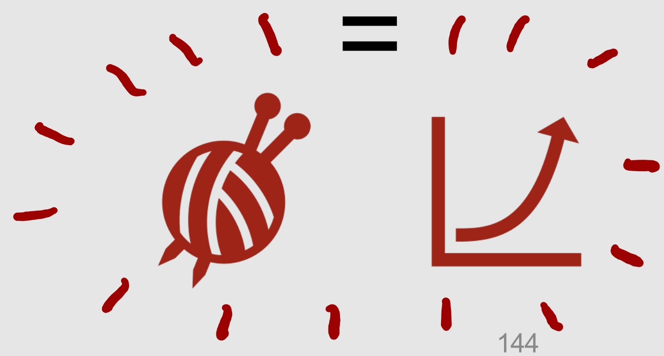
FrameNet

- Semantic role labeling project where roles are specific to frames rather than individual verbs
- **Frame:** A set of background information that unites a group of words

[$_{Arg1}$ The price of yarn] increased [$_{Arg2}$ 500%].

[$_{Arg1}$ The price of yarn] rose [$_{Arg2}$ 500%].

There has been a [$_{Arg2}$ 500%] rise in [$_{Arg1}$ the price of yarn].



Frames

- Background knowledge structures that define:
  - Specific **frame elements** associated with a given topic
  - Predicates that use these frame elements
- **Frame element:** A frame-specific semantic role

Attention

Lexical Unit Index

Definition:

This frame concerns a Perceiver's state of readiness to process and consider impressions of a Figure within a Ground. It is often unknown to the Perceiver whether or not the Figure exists within the Ground. Alternatively, the Expressor may be expressed as showing signs of the Perceiver's state of attentiveness.

Legislator tells consumers to be ALERT to dioxin levels.

They demand an ATTENTIVE gaze, a careful accounting of parts.

FEs:

Core:

Expressor [] **Excludes:** Perceiver	An entity (or event) associated with a Perceiver that gives evidence for a Perceiver's attentiveness.
Figure []	The entity that the Perceiver is specifically focussing on within the Ground.
Perceiver [] **Semantic Type:** Sentient	The individual that pays attention to the Ground.

Non-Core:

Circumstances []	The situation within which the Perceiver is alert.
Degree [] **Semantic Type:** Degree	The amount of attention that the Perceiver is paying to the Figure or Ground.
Ground []	The sensory field or subset of a sensory field that the Perceiver is attending to.
Manner [] **Semantic Type:** Manner	Any description of the event which is not covered by more specific FEs, including epistemic modification (probably, presumably, mysteriously), force (hard, softly), secondary effects (quietly, loudly), and general descriptions comparing events

Frame-frame Relations:

Inherits from: State
Is Inherited by:
Perspective on:
Is Perspectivized in:
Uses:
Is Used by: Emotions_of_mental_activity, Perception_active, Searching_scenario
Subframe of:
Has Subframe(s):
Precedes:
Is Preceded by:
Is Inchoative of:
Is Causative of:
See also:

Lexical Units:

alert.a, attend.v, attention.n, attentive.a, close.a, closely.adv, ignore.v, keep an eye.v

Frames

Core roles

- Frame-specific elements

Non-core roles

- More general elements
  - Time, location, etc.
- Similar to the ArgM arguments in PropBank

- Each word within a sentence or clause is understood to evoke a frame, and participate in that frame in some way
- FrameNet includes:
  - Manually specified frames and frame elements
  - Example sentences

Example Sentences

Frame: **change_position_on_a_scale**

[ITEM Oil] rose [ATTRIBUTE in price] [DIFFERENCE by 2%].

a steady increase [INITIAL VALUE from 9.5] [FINAL VALUE to 14.3] [ITEM in dividends]

[ITEM It] has increased [FINAL STATE to having them 1 day a month].

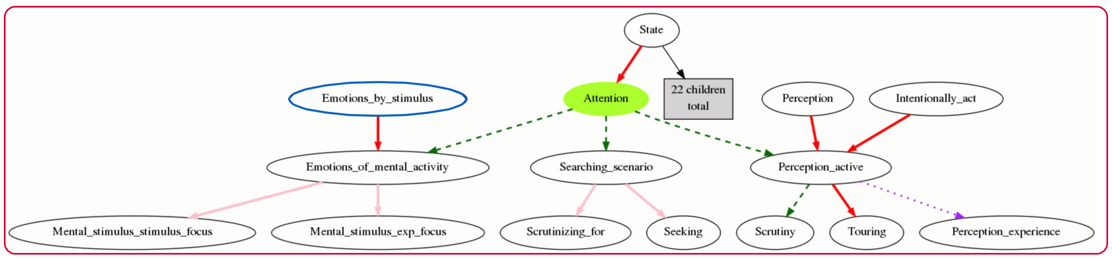
[ITEM Microsoft shares] fell [FINAL VALUE to 7 5/8].

[ITEM Colon cancer incidence] fell [DIFFERENCE by 50%] [GROUP among men].

a [DIFFERENCE 5%] [ITEM dividend] increase...

FrameNet

- Frame relationships (i.e., inheritance or causation) allow us to understand common event semantics across verbal and nominal causative and non-causative uses
- FrameNet databases have been developed for a variety of languages
- Link:
  - https://framenet.icsi.berkeley.edu/fndrupal/
- Manual:
  - Josef Ruppenhofer, Michael Ellsworth, Miriam R. L Petruck, Christopher R. Johnson, Collin F. Baker, Jan Scheffczyk: FrameNet II: Extended Theory and Practice (Revised November 1, 2016.): https://framenet2.icsi.berkeley.edu/docs/r1.7/book.pdf



This Week's Topics

Dependency Structure

Transition-Based Dependency Parsing

Graph-Based Dependency Parsing

Meaning Representations

Tuesday

Thursday

Model-Theoretic Semantics

First-Order Logic

Semantic Roles

Semantic Role Labeling

Selectional Preferences

Semantic Role Labeling

- **Semantic role labeling:** Automatically assigning semantic roles to predicate arguments
- Often solved using supervised machine learning methods

The University of Illinois Chicago offered free flu shots.

How are roles defined?

- Depends on the resource!
- Often, FrameNet and/or PropBank are used to:
  - Specify predicates
  - Define roles
  - Provide training and test data

Numerous approaches have been used to perform semantic role labeling.

- Feature-based algorithms:
  - Parse the input string
  - Traverse the parse to find predicates
  - Decide the semantic role (if any) of each node in the parse tree with respect to each predicate
- Feature-based algorithms employ standard supervised machine learning algorithms and a wide variety of feature representations
- Many approaches also perform a second pass to address **global consistency** using the Viterbi algorithm or reranking approaches
  - Constituents in FrameNet and PropBank cannot overlap
  - PropBank does not allow multiple arguments of the same type

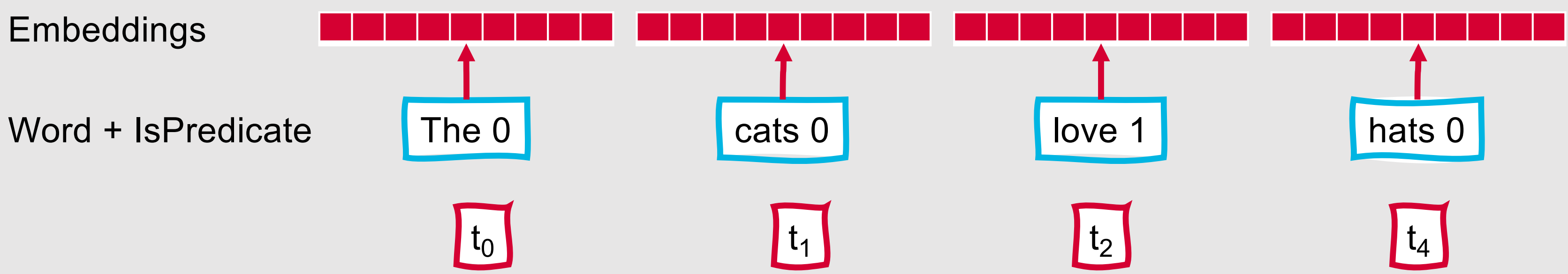
Features for Semantic Role Labeling

- Common features:
  - Governing predicate
  - Constituent type
  - Head word of the constituent
  - Part of speech of the head word
  - Path in the parse tree from the constituent to the predicate
  - Whether the voice of the surrounding clause is active or passive
  - Whether the constituent appears before or after the predicate
  - Set of expected arguments for the verb phrase
  - Named entity type of the constituent
  - First and last word(s) of the constituent

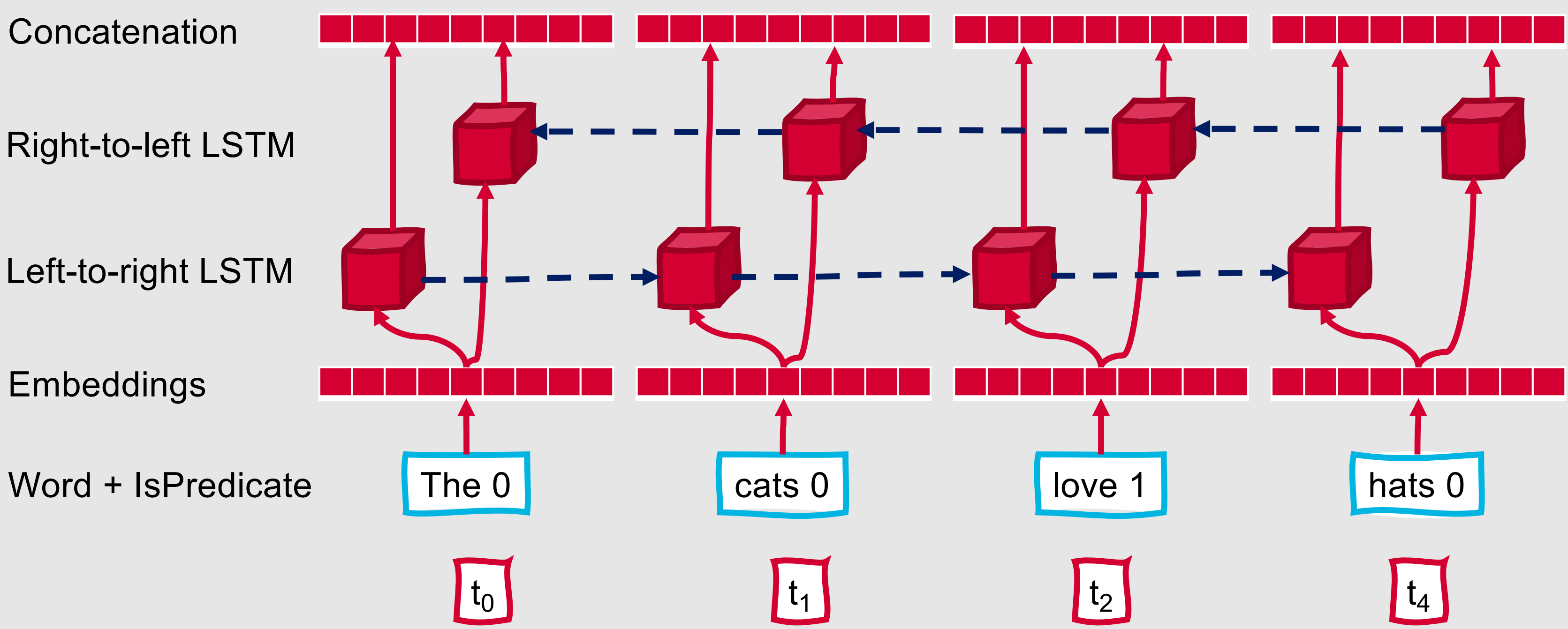
Modern SRL is also often performed using neural models.

- Frame SRL like other sequence labeling tasks
  - Given a predicate, detect and label spans with semantic roles
  - Use BIO tagging for this process
- Goal: Compute the highest probability tag sequence $\hat{y}$, given an input sequence of words w:
  - $\hat{y} = \underset{y \in T}{\operatorname{argmax}} P(y|w)$
- Global optimization can be addressed by applying Viterbi decoding directly to the softmax output

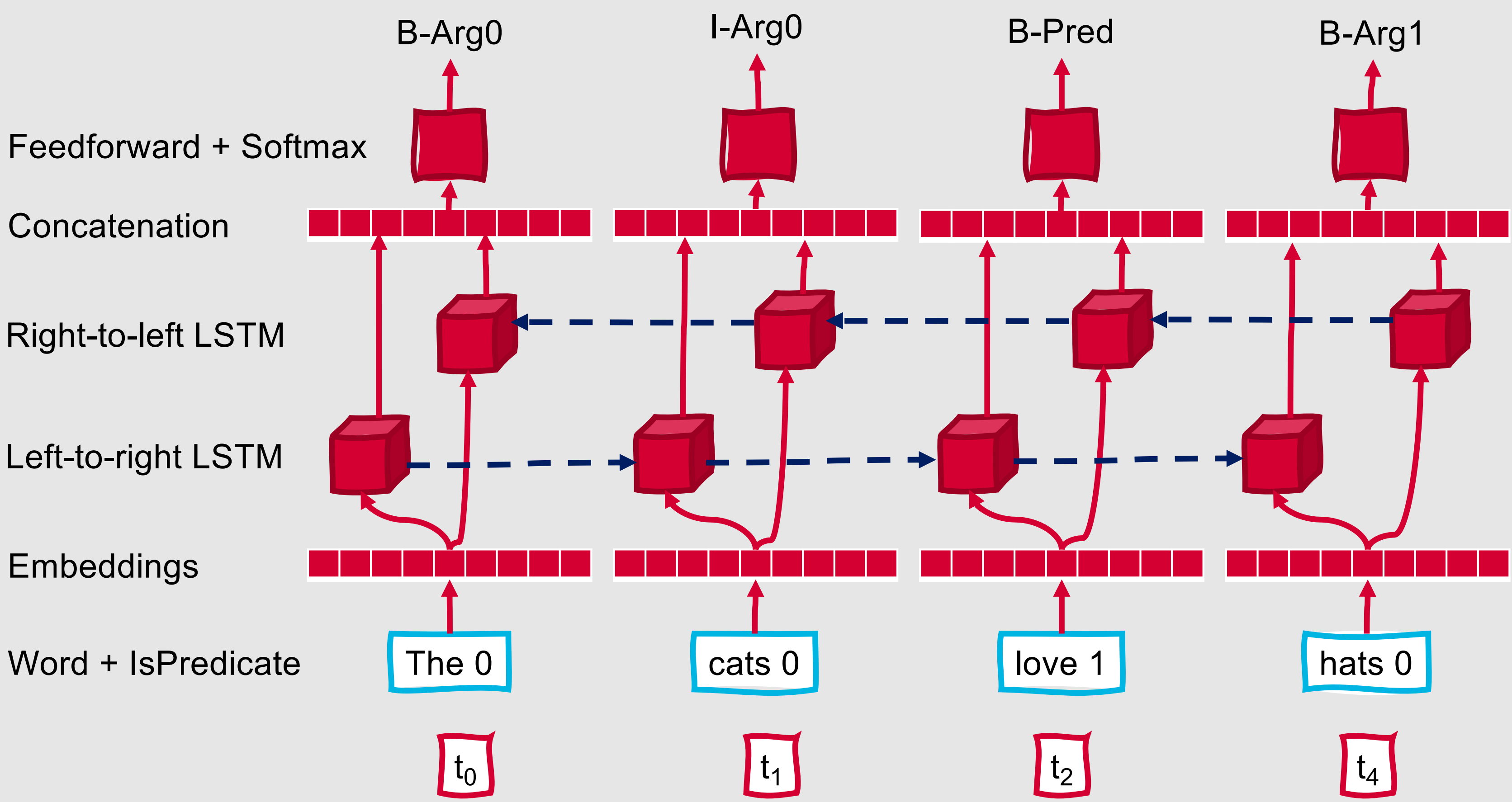
Neural Semantic Role Labeling



Neural Semantic Role Labeling



Neural Semantic Role Labeling



Evaluating Semantic Role Labelers

- **True positives:** Argument labels assigned to the correct word sequence or parse constituents
- Then, we can compute our standard NLP metrics:
  - Precision
  - Recall
  - F-measure

This Week's Topics

Tuesday

- Dependency Structure
- Transition-Based Dependency Parsing
- Graph-Based Dependency Parsing
- Meaning Representations

Thursday

- Model-Theoretic Semantics
- First-Order Logic
- Semantic Roles
- Semantic Role Labeling
- Selectional Preferences

Relationships between predicates and arguments can also be defined in other ways.

- Sometimes, there are conceptual or semantic limitations on which words can act as arguments to predicates
- We refer to these as **selectional restrictions**

What are selectional restrictions?

- **Selectional restrictions:** Semantic constraints placed upon predicates, governing the types of concepts that can fill those predicates' semantic roles

Selectional Restrictions

- Associated with senses, not words themselves
- Vary in their specificity
  - To eat: THEME should be edible
  - To sip: THEME should be edible and liquid
- The set of concepts needed for representing selectional restrictions is open-ended
  - Being a liquid
  - Being edible
  - …
- This makes selectional restrictions different from other ways to represent lexical knowledge
  - For example, parts of speech are finite and limited

One way to represent selectional restrictions….

- Extend the logical representations we've already seen
  - Use the same components we've used for representing events
    - Event variable
    - Predicate denoting event
    - Variables and relations for event roles

Representing Selectional Restrictions

$$\exists e, x, y \ \mathrm{Eating}(e) \land \mathrm{Agent}(e, x) \land \mathrm{Theme}(e, y)$$

$$\exists e, x, y \ \mathrm{Eating}(e) \land \mathrm{Agent}(e, x) \land \mathrm{Theme}(e, y) \land \mathrm{EdibleThing}(y)$$

$$\exists e, x, y \ \mathrm{Eating}(e) \land \mathrm{Eater}(e, x) \land \mathrm{Theme}(e, y) \land \mathrm{EdibleThing}(y) \land \mathrm{Pizza}(y)$$

Selectional Preferences

- **Selectional restrictions** → hard constraints
- **Selectional preferences** → soft constraints
- Many systems tend to use selectional preferences rather than selectional restrictions

She was way faster than everyone else …the other runners **ate her dust**.

Spit that out, you **can't eat plastic**!

Selectional Preference

- Selectional preferences, $S_P(v)$, are defined as the difference between two distributions:
  - Distribution of the expected semantic classes, $P(c)$
  - Distribution of the expected semantic classes for a specific verb, $P(c|v)$
- This difference can be quantified using **Kullback-Leibler (KL) divergence**, $D(P||Q)$:
  - $D(P||Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)}$
  - $S_P(v) = D(P(c|v)||P(c)) = \sum_c P(c|v) \log \frac{P(c|v)}{P(c)}$
- **Selectional association** then indicates how much a given class contributes to a verb's overall selectional preference
  - $A_R(v, c) = \frac{1}{S_P(v)} P(c|v) \log \frac{P(c|v)}{P(c)}$

Selectional Preference via Conditional Probability

- We can also model selection preference strength using conditional probability
  - Probability of an argument noun *n* given a predicate verb *v* for a particular relation *r*
- Can be computed using log co-occurrence frequency or modified maximum likelihood estimates
  - $P(v|n,r) = \begin{cases} \frac{C(n,v,r)}{C(n,r)} \text{ if } C(n,v,r) > 0 \\ 0 \text{ otherwise} \end{cases}$

How do we evaluate the quality of calculated selectional preferences?

Pseudoword task

- Determine which of two words are more preferred by a given verb, and compute how often the selectional preference model makes the correct choice

Human selectional preference scores

- Check correlation between human selectional preference scores and those predicted by the model

Summary: Model-Theoretic Semantics and Semantic Role Labeling

- In **model-theoretic semantics**, the model serves as a formal construct representing a particular state of affairs in the world
- **First-order logic** maps linguistic input to world knowledge using logical rules
- First-order logic makes use of both **existential** and **universal** quantifiers
- **Description logic** models semantic domains using subsets of first-order logic, restricting expressiveness such that it guarantees the tractability of certain kinds of inference
- **Semantic roles** define argument roles with respect to a predicate
- **PropBank** and **FrameNet** also define various general and specific semantic role types
- **Semantic role labeling** is the task of automatically assigning semantic roles to words or spans of words in a specific context
- **Selectional restrictions** are hard constraints placed upon the semantic properties of arguments
- **Selectional preferences** are soft constraints placed on those properties, and can have varying **selectional association** strength