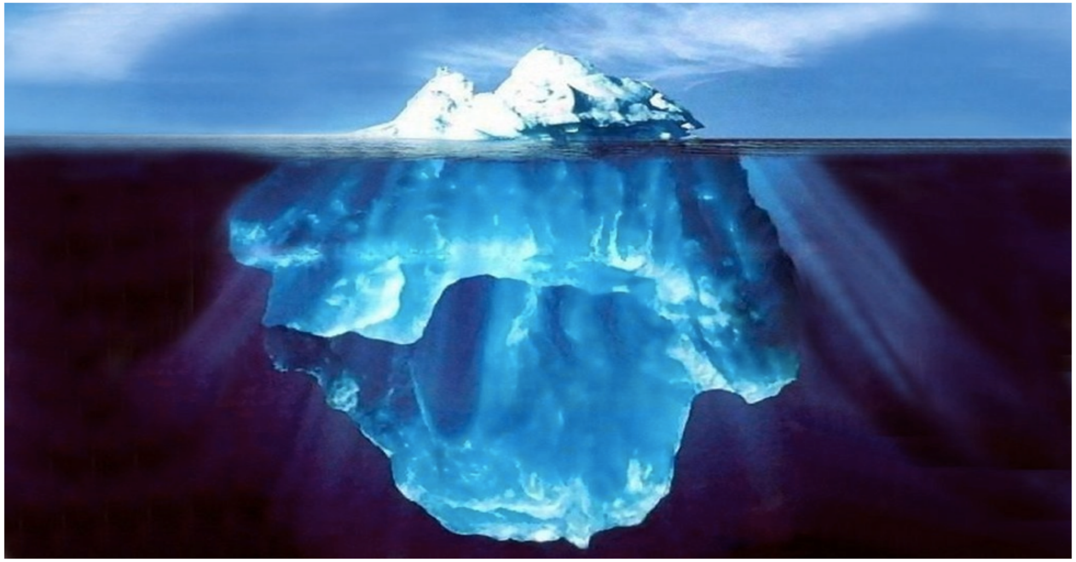
ML4NLP
Introduction to Syntax and Parsing



Dan Goldwasser

Purdue University

dgoldwas@purdue.edu

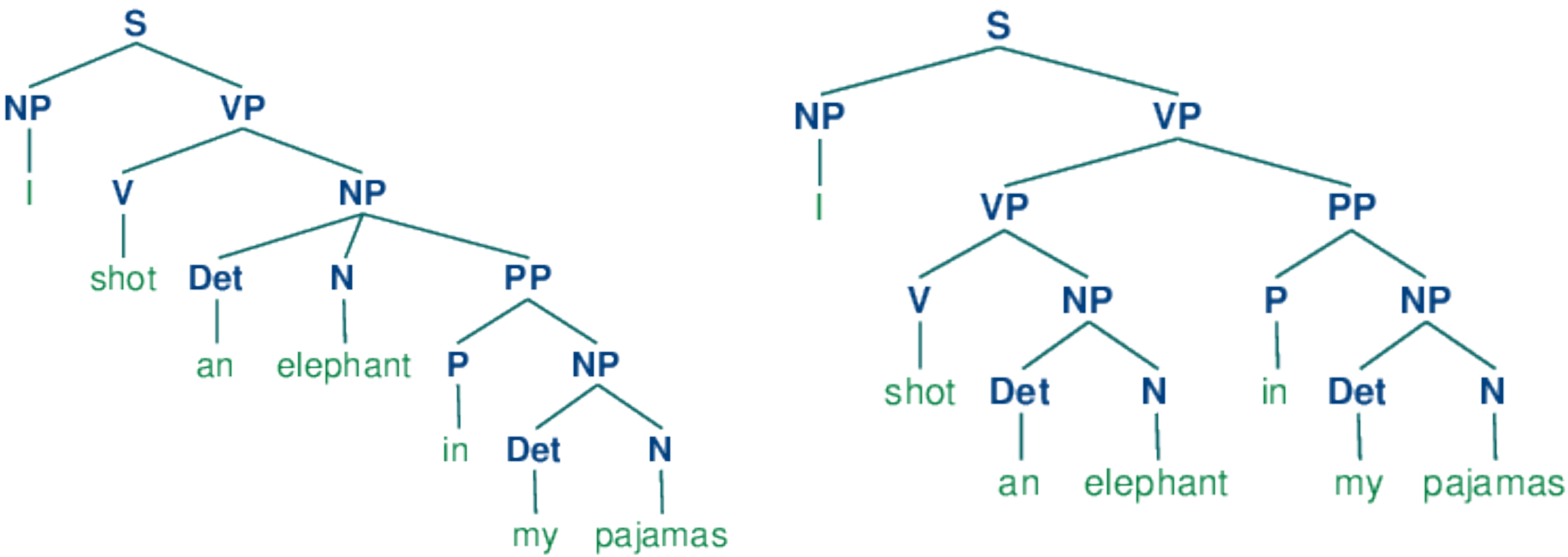
CS 590NLP

“I shot an elephant in my pajamas”

“How he got into my pajamas, I'll never know.”

Groucho Marx

"I shot an elephant in my pajamas."



"I shot an elephant in the zoo."

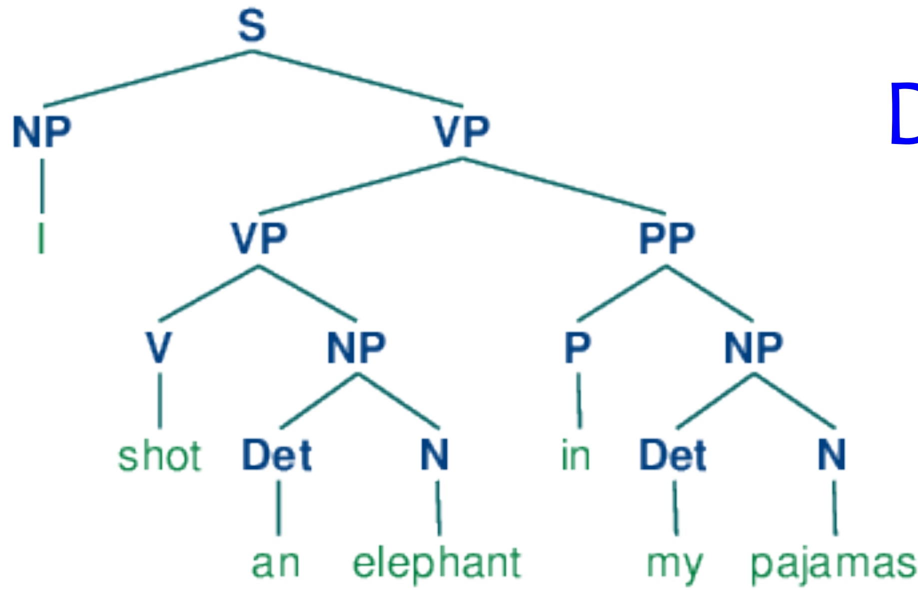
Parsing

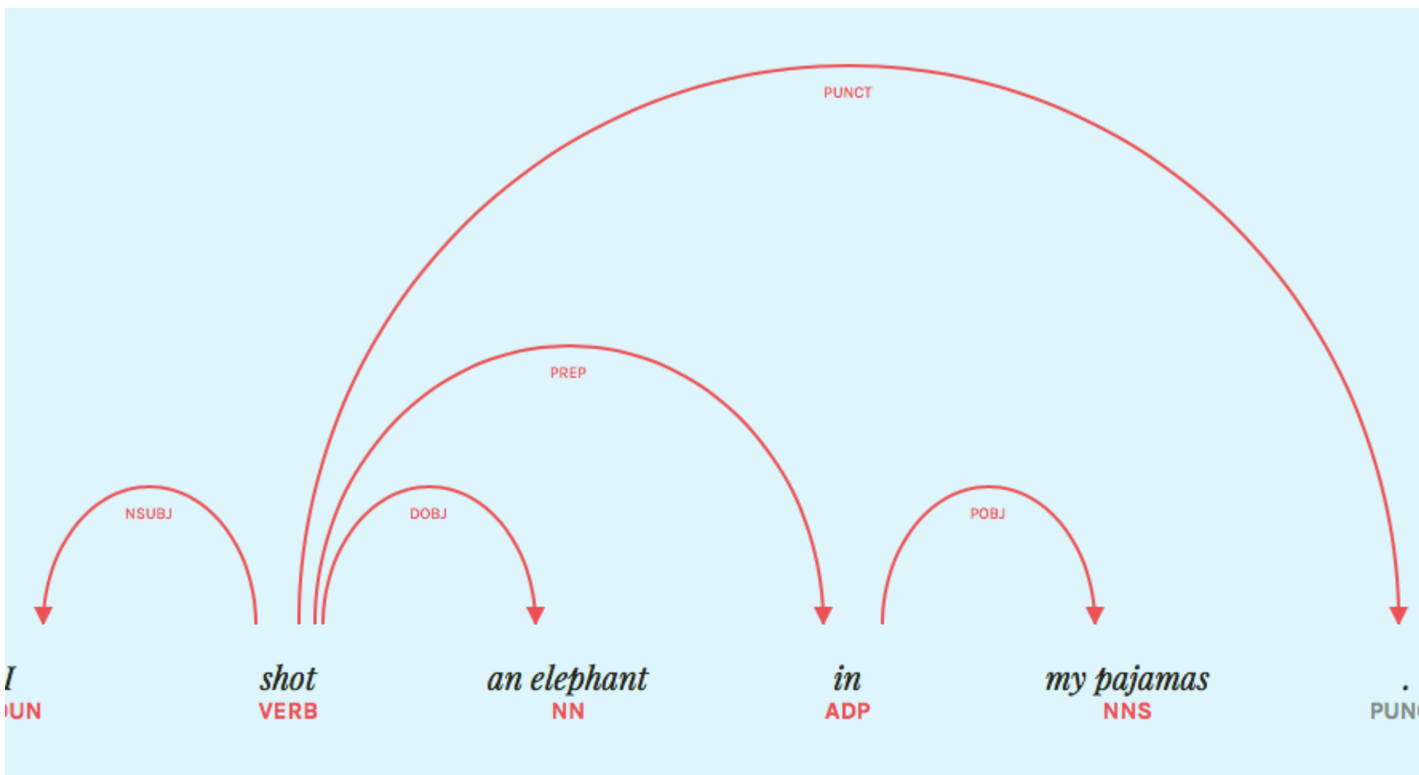
Language is not just a stream of words,

We want to represent linguistic structure!

- *Two views:*
  - ***Constituency Parsing*****:** *Build a hierarchical phrase structure*
  - ***Dependency Parsing:*** *Show words dependencies (dependency = modifiers, or arguments)*

Dependency and constituent parsing





Constituency Parsing

*"the good old days..": **Write a program!***

S → NP VP
NP → Det N
NP → NP PP
VP → V NP
VP → VP PP
PP → P NP

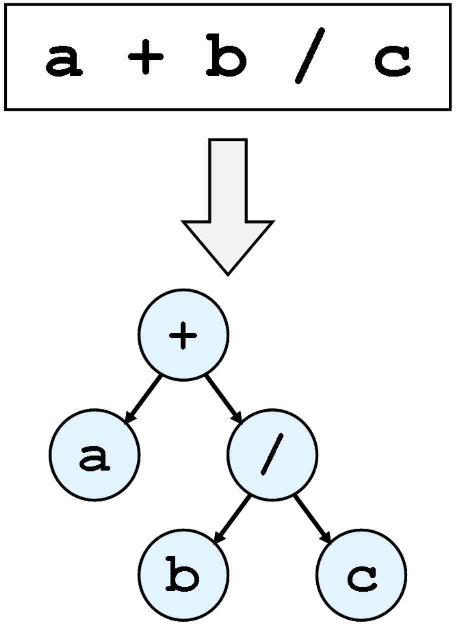
NP → John
NP → Mary
N → binoculars
N → dog
V → saw
P → with
Det → a

Can you parse: "Mary saw John with binoculars"?

How about: "Mary saw a dog with binoculars"?

Constituency Parsing

- *"the good old days..": **Write a program!***
- *Can you treat natural and formal languages in the same way?*



Constituency Parsing

“Fed raises interests rates 0.5% in effort to control inflation”

How many parsing options?

Million of possible parses in a broad-coverage grammar

This explains the popularity of statistical methods in NLP : millions of options, but only a few are **likely**!

CFG

- **Formally**: a context-free grammar is:
- G = (T,N,S,R)
  - T: terminal symbols
  - N: non-terminals
  - S: start symbol
  - R: production rules X→ Y *(where X is N, Y is T or N)*
- A grammar G generates a language L

PCFG

- **Formally**: a **probabilistic** context-free grammar:
- G = (T,N,S,R,**P**)
  - T: terminal symbols
  - N: non-terminals
  - S: start symbol
  - R: production rules X→ Y *(where X is N, Y is T or N)*
  - **P: probability function over R**

$$\forall X \in N, \sum_{X \to Y \in R} P(X \to Y) = 1$$

PCFG example

S → NP VP	1.0
NP → Det N	0.6
NP → NP PP	0.4
VP → V NP	0.6
VP → VP PP	0.4
PP → P NP	1.0

NP → John	0.3
NP → Mary	0.3
N → binoculars	0.2
N → dog	0.2
V → saw	0.2
P → with	0.4
Det → a	0.4
…	

P(Tree) – The probability of a tree is the product of the probabilities of the rules used to generate it.

Constituency Parsing as Structured Learning

- Can you define PCFG as a structured prediction problem?
  - How would you define the prediction problem?
  - What are the dependencies in the model?
  - What are the parameters you need to learn?
  - What are good features?

CKY Algorithm

- Dynamic programming algorithm for parsing
- Given a CFG **G** and a string **w,** *determine can **G** parse **w**?*
- We assume **G** is a CNF:
  - *Each rule has at most 2 symbols on the right $A \rightarrow BC$ or $A \rightarrow B$, $A \rightarrow a$*
- The algorithm maintains a triangular DP table.
  - Bottom row: parse strings of size 1
  - Second bottom row: parse strings of size 2..
  - Top row: parse the entire sentence!

DP Triangular Table

$X_{1,5}$				
$X_{1,4}$	$X_{2,5}$			
$X_{1,3}$	$X_{2,4}$	$X_{3,5}$		
$X_{1,2}$	$X_{2,3}$	$X_{3,4}$	$X_{4,5}$	
$X_{1,1}$	$X_{2,2}$	$X_{3,3}$	$X_{4,4}$	$X_{5,5}$
w_1	w_2	w_3	w_4	w_5

Table for string '**w**' that has length 5

Constructing The Triangular Table

S → AB | BC
A → BA | a
B → CC | b
C → AB | a

{B}	**{A, C}**	**{A, C}**	**{B}**	**{A, C}**
b	**a**	**a**	**b**	**a**

At each point consider possible rules,
and their probabilities

S						
	VP					
S						
	VP			PP		
S		NP			NP	
NP	V, VP	Det.	N	P	Det	N
she	eats	a	fish	with	a	fork

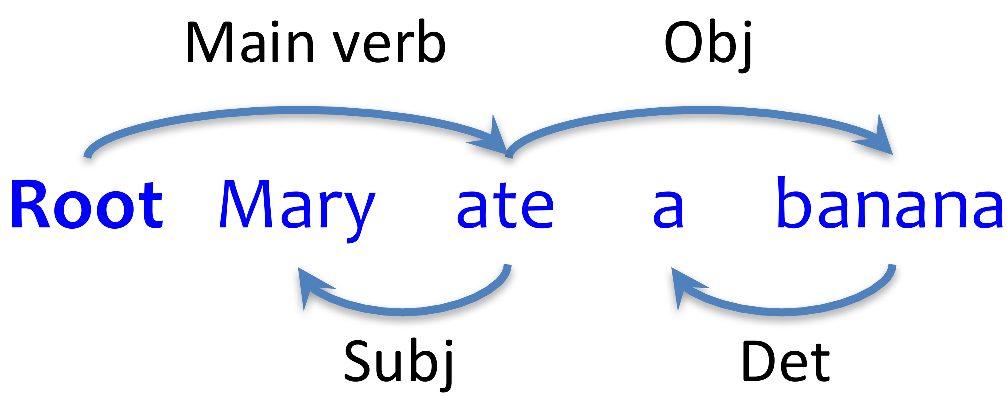
CYK Algorithm

- Similar to Viterbi, keep backpointers to reconstruct the parse tree from the table
- The rule activation scoring function depends on the dependency assumptions
  - Look at the probability of previous row activation, and consider the conditional probability of the rule given previous parses.
- Overall Complexity: $O(n^3 G)$

Option 2: **dependency parsing**

- **Key idea**: syntactic structure represented as relations between lexical items, called *dependencies*

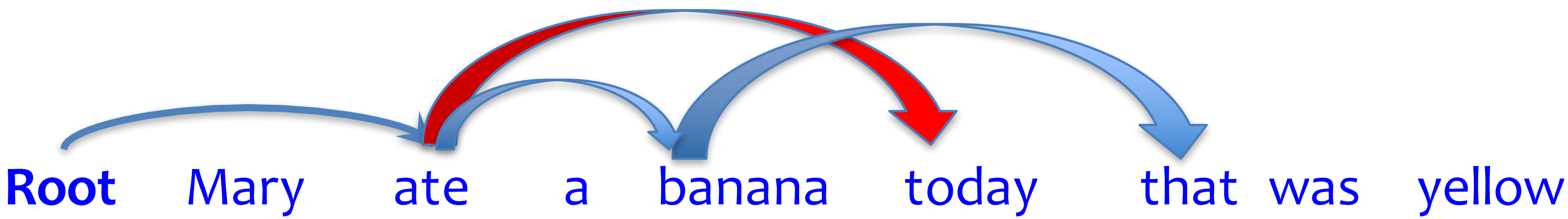
Dependencies can be represented as a graph, where the nodes are words, and edges are dependencies, which are: (1) directional (2) often **typed**



Non-Projective Structure

Projective structure: *no crossing edges*

Are those really needed?



However, we will often assume non-projectivity.

- *It makes life easier*
- *It doesn't occur often*

Dependency Parsing

- We need to answer two questions –
  - How can you make parsing decisions? (i.e., *inference*)
  - How do you learn the parameters to score these decisions?

Dependency Parsing

- **Parser**: for each word, choose which *other* word it depends on.
  - *You can choose to label these dependencies*
- **Constraints**:
  - Only one root
  - No cycles

➔ *Essentially, force a tree structure*

- **Additional Constraints:** no crossing dependencies

Parsing Approaches

- Two competing approaches –
- Exact Inference: mostly graph based algorithms (e.g., spanning tree) but also ILP
- Approximate inference: *linear time* transition parser
- **Transition based parser are very popular!**

Greedy Transition-based Dependency Parsing (Nivre'03)

- Parser operates by maintaining two data structures:
  - Stack and Buffer
- Parsing is done via a sequence of operations.
  - Pushing the words from the buffer to the stack, and associating dependency edges over words in the stack.
  - **Shift**: take a word from the top of the buffer, and put it on top of the stack.
  - **Left/Right Arc**: Dependency operations associate dependencies between words in the stack, and remove the dependent word from the stack.
- Parsing sequence ends when the stack and buffer are empty

Stack	Buffer
[Root]	I like lettuce
shift	
[Root] I	like lettuce
shift	
[Root] I like	lettuce
Left Arc	
[Root] like	lettuce
Shift	
[Root] like lettuce	
Right Arc	
[Root] like	
Right Arc	
[Root]	

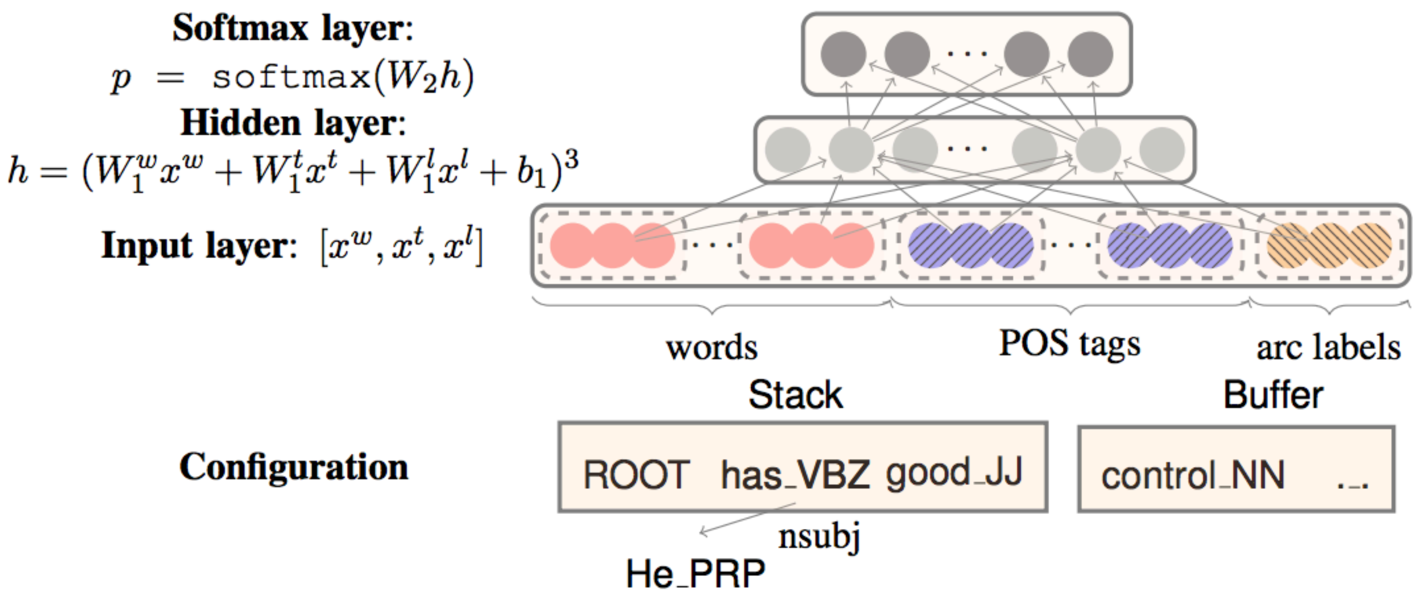
Learning for Dependency Parsing

- Learning a transition parser: use data to build a scoring function for parser operations.
  - **This should sound familiar..**
- Break the data into a sequence of decisions, and train a ”next-state” function.
  - *Local learning, (greedy) inference only at test time.*
- Traditionally: SVM, LR,..
  - Essentially a multiclass classifier over the current state of the parser.

Learning for Dependency Parsing

- Which features would you consider?

Deep Learning for Dependency Parsing (Chen, Manning'14)




Ceci n'est pas une pipe.

From Syntax to Semantics

- The syntactic structure of the sentence captures some semantic properties (e.g., recall the PP attachment problem).
- However, it does not account for meaning in a broad sense.
- Interesting question: What is a computational model for *meaning*?

What is the meaning of meaning?

Semantic

- We distinguish between:

- **Lexical semantics**: meaning of words

- **Compositional semantics**: Combine individual units to form the meaning of larger units.

Applications

- Semantics is what we really care about:
  - Question answering
  - Intelligent information access
  - Robot communication
  - Summarization
  - …

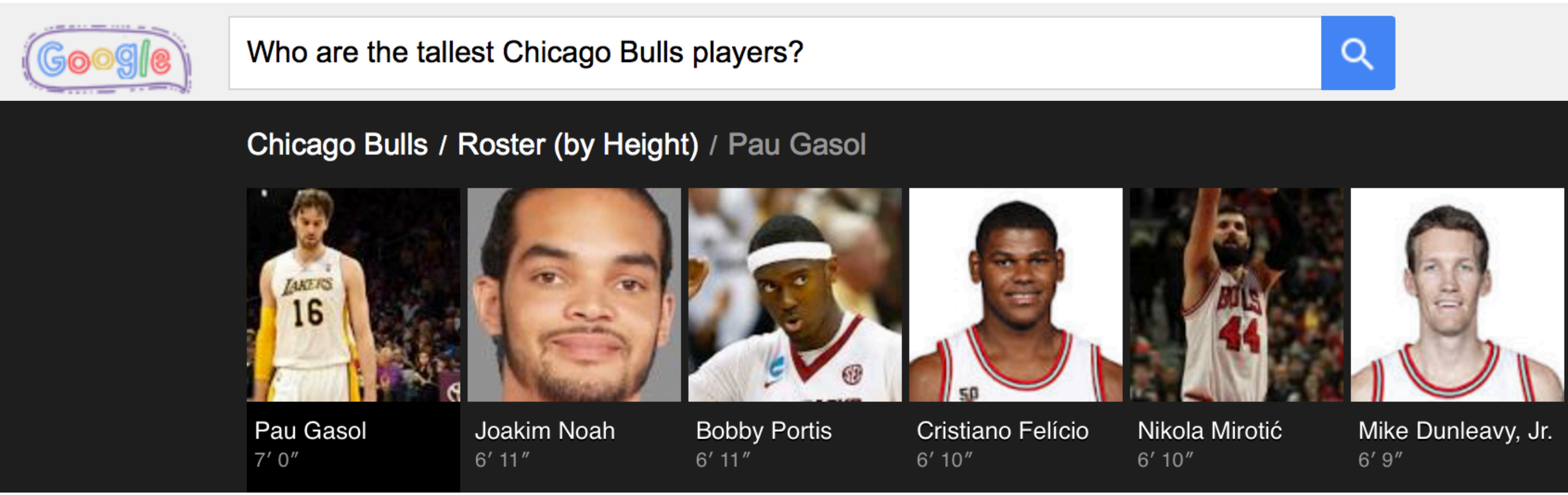
Deep vs. Shallow Semantics

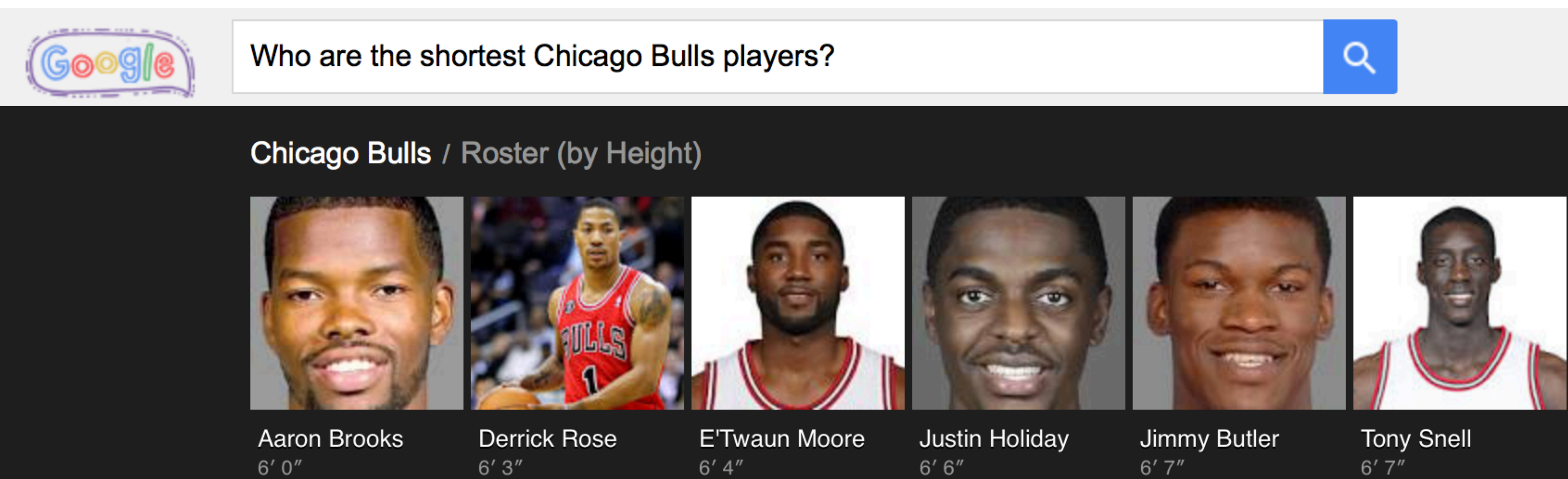
- Surprisingly, we tend to believe that dogs understand much more!
- Similarly – shallow NLP performs surprisingly well!

Semantics

- We will look at two semantics problems:
  - **Formal Semantic Representation**: *find a mathematical representation of meaning*
  - **Information Extraction**: “machine reading” view- populate a DB of facts from text .

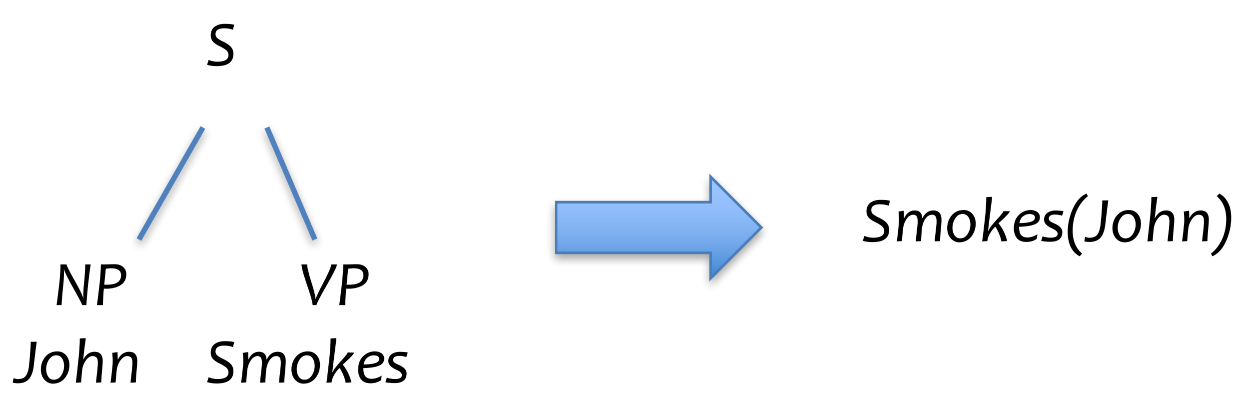
Formal Models of Meaning





Formal Models of Meaning

- Formal model for compositional semantics:
  - Form the semantics of parents, based on the semantics of the children
- *We assume a dictionary of items:*
  - **Constant** symbols
  - **Functions**



Constants and Functions

- **Constants**
  - Purdue University
  - Barak Obama
- **Properties:**
  - Red (x), Small (x),..
- **Relations:**
  - Love(x,y), PresidentOf(Barak Obama, USA)

Generating a meaning representation

- We assume that syntactic representations and compositional semantics are highly dependent
- **Simple algorithm:**
  - Create a parse tree
  - Find semantic representation of words (leaf nodes)
  - Combine semantics of children into parent node (bottom up)

Semantic Parser

- **Key idea**: *augment syntactic parsing with meaning!*

S → NP VP	NP → John
NP → Det N	NP → Mary
NP → NP PP	N → binoculars
VP → V NP	N → dog
VP → VP PP	V → saw
PP → P NP	P → with
	Det → a

Semantic Parser

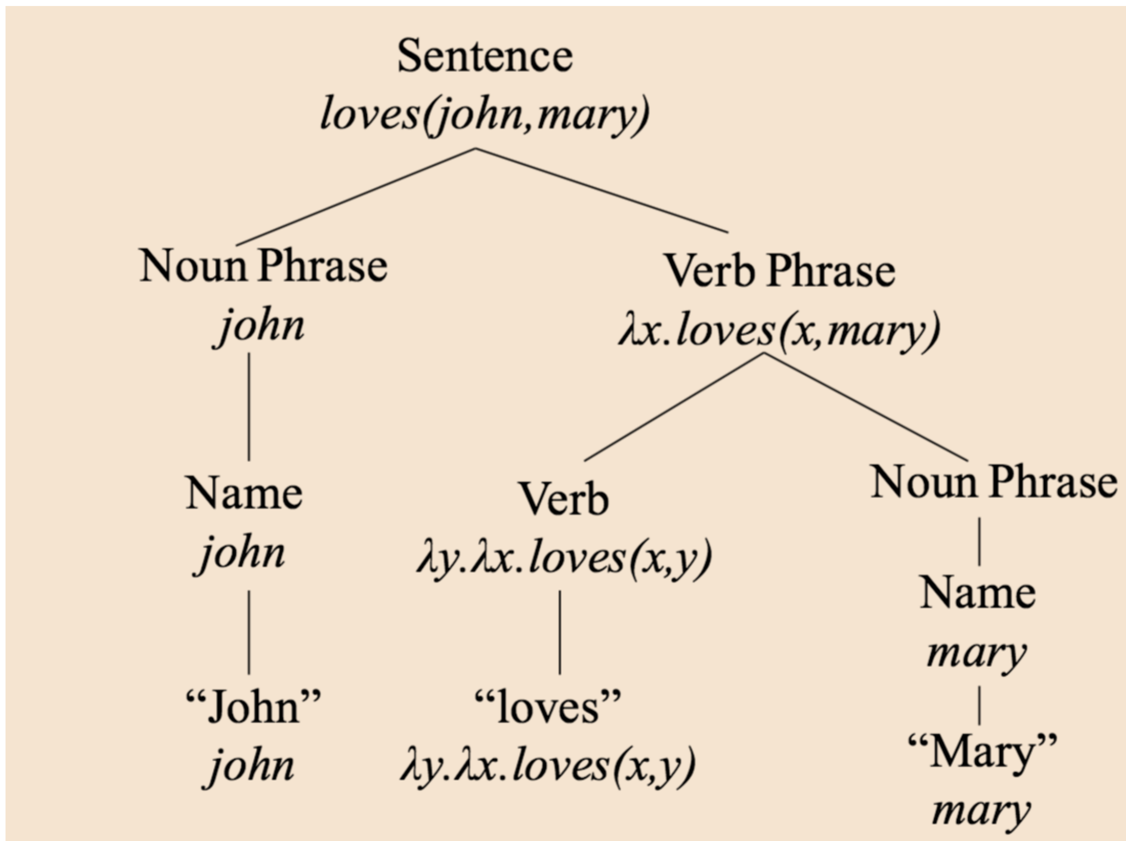
- **Key idea**: *augment syntactic parsing with meaning!*

S [SM] → NP [NPM] VP [VPM], Apply(SM, NPM,VPM)
VP [VPM] → V [VM] NP [NPM], Apply(VPM, VM,NPM)
NP [NPM] → N [NM], Apply(NP, N)
..

N [john]→ John
N [mary]→ Mary
V [λx.saw(x)] → saw

This is sometimes called a lexicon

Generating a meaning representation



Learning for Semantic Parsing

- Similar to syntactic parsing, there are many possible meaning derivation for a single sentence
  - **Each could result in a different semantic representation!**
- To help us disambiguate the meaning of a sentence, we can define probabilistic parser:

N [john]→ John 0.4
N [mary]→ Mary 0.2
V [λx.saw(x)] → saw 0.9

Grounded Language Interpretation

- Compositionality: constructing meaning by composing the meaning of lower level units.
  - Lowest level (“leaves”) are typically constant symbols
- **Where do the symbols come from?**
- We assume a world model, providing the relevant set of symbols
  - People on your smartphone, transactions in a DB, entities on wikipedia, real world objects (”pick up that block”)
- **Analyzing the difficulty of semantic interpretation:**
  - Complexity of the input language, complexity of the set of symbols, complexity of their mapping

Grounded Language Interpretation

- ”pick up the green piece”
- ”pick up the green piece that’s next to the blue piece”
- ”pick up the green piece that’s shaped like a lettuce”
- “pick up the green piece that’s at the left end of the bottom row.

A Joint Model of Language and Perception for Grounded Attribute Learning. Matuszek et-al. 2012

Grounded Language Interpretation

- Create a meaning representation capturing the mapping from language to precepts in the real world

this	red	block	is	in the	shape	of a	half-pipe
N/N	N	$N\backslash N$	$S\backslash N/N$	N/N	N/NP	NP/NP	NP
$\lambda f.f$	$\lambda x.color(x, red)$	$\lambda f.f$	$\lambda f.\lambda g.\lambda x.f(x) \wedge g(x)$	$\lambda f.f$	$\lambda y.\lambda x.shape(x, y)$	$\lambda x.x$	$arch$

$$\frac{N: \lambda x.color(x, red) \quad N\backslash N: \lambda f.f}{N: \lambda x.color(x, red)}$$

$$\frac{N/N: \lambda f.f \quad N: \lambda x.color(x, red)}{N: \lambda x.color(x, red)}$$

$$\frac{N/N: \lambda f.f \quad N/NP: \lambda y.\lambda x.shape(x, y)}{N/NP: \lambda y.\lambda x.shape(x, y)}$$

$$\frac{NP/NP: \lambda x.x \quad NP: arch}{NP: arch}$$

$$\frac{N/NP: \lambda y.\lambda x.shape(x, y) \quad NP: arch}{N: \lambda x.shape(x, arch)}$$

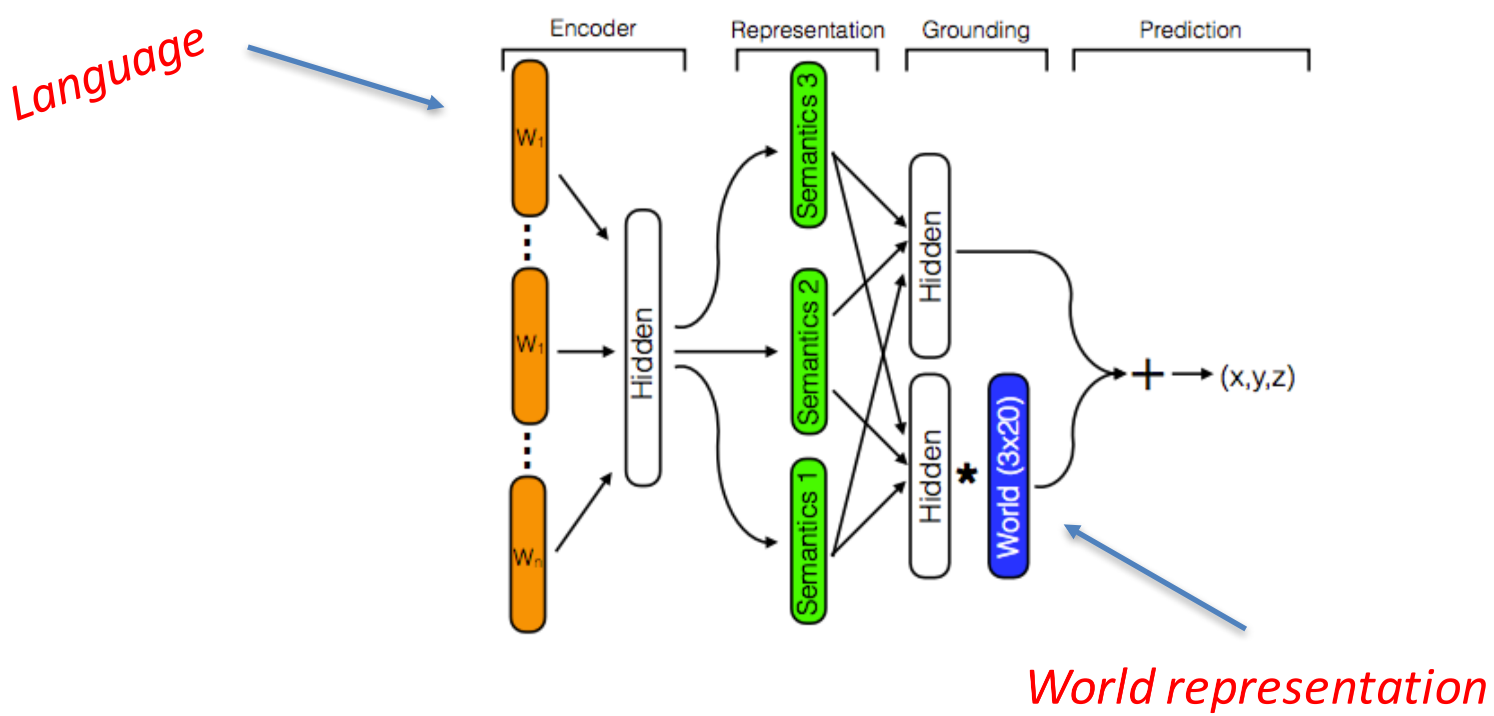
$$\frac{S\backslash N/N: \lambda f.\lambda g.\lambda x.f(x) \wedge g(x) \quad N: \lambda x.shape(x, arch)}{S\backslash N: \lambda g.\lambda x.shape(x, arch) \wedge g(x)}$$

$$\frac{N: \lambda x.color(x, red) \quad S\backslash N: \lambda g.\lambda x.shape(x, arch) \wedge g(x)}{S: \lambda x.shape(x, arch) \wedge color(x, red)}$$

The (probability of) truth value of these predicts depends on real world grounding

Grounded Language Interpretation

- Create a scoring function, connecting the two representations



Natural Language Communication with Robots. Bisk et-al.2016

Grounded Language Interpretation

- Two competing approaches.
  - Create an explicit meaning representation
  - Create a scoring function that ranks meaning representations (or their outcomes)
- We discussed it in the context of grounded representations ("real world objects")
  - Similar discussion for different settings (e.g., DB access).
- Which one will be easier to learn? What kind of supervision effort is needed in either?

Scaling up

Voters go to the polls in four states on Tuesday, with Michigan the biggest prize for both parties.

Donald J. Trump seeks to strengthen his position as the Republican front-runner, while his rivals look to slow his drive toward the nomination.

For the Democrats, Senator Bernie Sanders of Vermont faces a crucial test in his upstart campaign to derail Hillary Clinton.

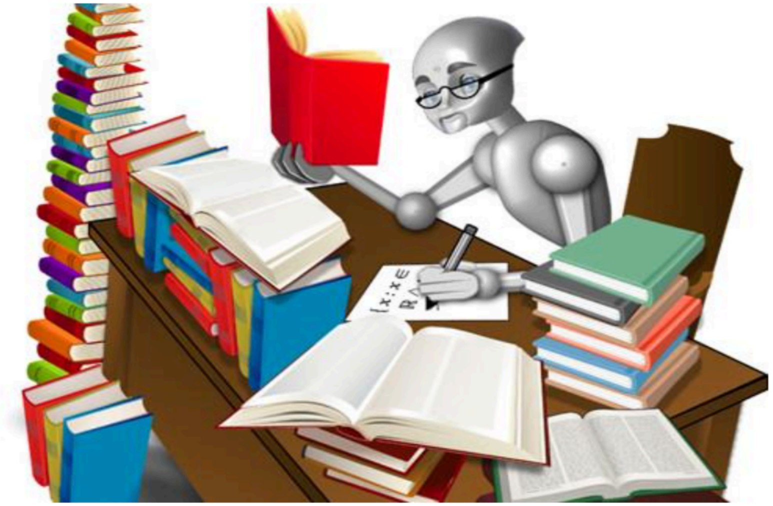
Here are some of the things we will be watching in the contests in Hawaii, Idaho, Michigan and Mississippi.

NYTimes article

Machine Reading

- More realistic task: *given unstructured text, create structured knowledge*

- Simple Examples:
  - *Named Entity Recognition*
- *More complicated:*
  - *Relationships between entities*



IE Example

Bernie Sanders **is-from** Vermont

Bernie Sanders **is-a** democrat

For the Democrats, Senator Bernie Sanders of Vermont faces a crucial test in his upstart campaign to derail Hillary Clinton. Here are some of the things we will be watching in the contests in Hawaii, Idaho, Michigan and Mississippi.

Relation Extraction

- We make a distinction between *closed* and *open* IE
- Closed: focus on a small set of relations
  - Easy to think about as a supervised task
- Open: find all relations

Relation Extraction

- Popular task: ACE 2003 defined 4 types:
  - **Role**: member, owner, affiliate, client
  - **Part**: subsidiary, physical part-of, set membership
  - **At**: location, based-in, residence
  - **Social**: parent, sibling, spouse
- **Realistic settings**: *Freebase has thousands of relations!*

Building Relation Extractors

- Simple pattern recognition

Hearst (1992)

Agar is a substance prepared from a mixture of red algae, such as Gelidium, for laboratory or industrial use.

What does Gelidium mean?

Pattern based Relation Extraction

Y such as X ((, X)* (, and/or) X)
such Y as X…
X… or other Y
X… and other Y
Y including X…
Y, especially X…

Hearst, 1992. Automatic Acquisition of Hyponyms.

Pattern based Relation Extraction

Hearst pattern	Example occurrences
X and other Y	...temples, treasuries, and other important civic buildings.
X or other Y	bruises, wounds, broken bones or other injuries...
Y such as X	The bow lute, such as the Bambara ndang...
such Y as X	...such authors as Herrick, Goldsmith, and Shakespeare.
Y including X	...common-law countries, including Canada and England...
Y, especially X	European countries, especially France, England, and Spain...

Bootstrapping

- Simple idea:
  - Given a small seed set of relations (e.,g by mining patterns)
  - And A LOT of unsupervised text
  - *Find mentions of relation in the text*
  - *Use mentions to come up with new patterns!*
    - Grep/Google for "Mark Twain" and "Elmira"

      "Mark Twain is buried in Elmira, NY."

      → X is buried in Y

      "The grave of Mark Twain is in Elmira"

      → The grave of X is in Y

      "Elmira is Mark Twain's final resting place"

      → Y is X's final resting place

Supervised Relation Extraction

- Given a sentence, find the list of entities, and predict if there is a relation.
- **Key problem**: finding a good feature representation

Features	P	R	F
Words	69.2	23.7	35.3
+Entity Type	67.1	32.1	43.4
+Mention Level	67.1	33.0	44.2
+Overlap	57.4	40.9	47.8
+Chunking	61.5	46.5	53.0
+Dependency Tree	62.1	47.2	53.6
+Parse Tree	62.3	47.6	54.0
+Semantic Resources	63.1	49.5	55.5

Table 2: Contribution of different features over 43 relation subtypes in the test data

Zhou et al. 2005

Scaling up RE

- **Key problem**: realistic machine reading requires dealing with thousands of relations.
- Directly annotating for this task is not reasonable, how can we scale up?
- **Key idea: *distant supervision***
  - Similar to Bootstrapping + Learning

Distant Supervision

- Assume we have a collection of relations
  - Easy! (e.g., Freebase, Wikipedia,..)
- ..and that if two entities appear in a relation, sentences containing these two entities will express this relationship.
- ***Use such sentences as noisy training data!***

Distant Supervision Example


article | discussion | edit this page | history

Log in / create account

navigation
- Main page
- Contents
- Featured content
- Current events
- Random article

search

Go | Search

interaction
- About Wikipedia
- Community portal
- Recent changes
- Contact Wikipedia
- Donate to Wikipedia
- Help

toolbox
- What links here
- Related changes
- Upload file
- Special pages
- Printable version
- Permanent link
- Cite this page

languages
- Deutsch
- فارسی
- Français
- Bahasa Indonesia
- Italiano
- 日本語
- Norsk (bokmål)
- Polski
- Русский
- 中文

Lawrence Livermore National Laboratory

From Wikipedia, the free encyclopedia

Coordinates: 37.686024°N 121.709547°W

The **Lawrence Livermore National Laboratory** (**LLNL**) in Livermore, California is a scientific research laboratory founded by the University of California in 1952. It is funded by the United States Department of Energy (DOE) and managed by Lawrence Livermore National Security, LLC (LLNS), a partnership of the University of California, Bechtel Corporation, Babcock and Wilcox, the URS Corporation, and Battelle Memorial Institute. On October 1, 2007 LLNS assumed management of LLNL from the University of California, which had exclusively managed and operated the Laboratory since its inception 55 years before.

Contents [hide]

Lawrence Livermore National Laboratory

University of California
Lawrence Livermore National Laboratory

Motto	"Science in the national interest"
Established	1952 by the University of California
Research Type	National security, nuclear science
Budget	US$1.6 billion
Director	George H. Miller
Staff	6,800
Location	Livermore, California
Campus	3.2 km² (800 acres)
Operating Agency	Lawrence Livermore National Security, LLC
Website	www.llnl.gov

Background [edit]

LLNL is self-described as "a premier research and development institution for science and technology applied to national security."[1] Its principal responsibility is ensuring the safety, security and reliability of the nation's nuclear weapons through the application of advanced science, engineering and technology. The Laboratory also applies its special expertise and multidisciplinary capabilities to preventing the proliferation and use of weapons of mass destruction, bolstering homeland security and solving other nationally important problems, including energy and environmental security, basic science and economic competitiveness.

LLNL is home to many unique facilities and a number of the most powerful computer systems in the world, according to the TOP500 list, including Blue Gene/L, the world's fastest computer from 2004 until Los Alamos National Laboratory's Roadrunner supercomputer surpassed it in 2008. The Lab is a leader in technical innovation: since 1978, LLNL has received a total of 118 prestigious R&D 100 Awards, including

Aerial view of Lawrence Livermore National Laboratory

"The Lawrence Livermore National Laboratory (LLNL) in Livermore, California is a scientific research laboratory founded by the University of California in 1952."

↓

LLNL EQ Lawrence Livermore National Laboratory
LLNL LOC-IN California
Livermore LOC-IN California
LLNL IS-A scientific research laboratory
LLNL FOUNDED-BY University of California
LLNL FOUNDED-IN 1952