Sentence

Grammar

- Two views of linguistic structure
  - Constituency
    - context-free grammar (CFG)
  - Dependency

The Penn Treebank

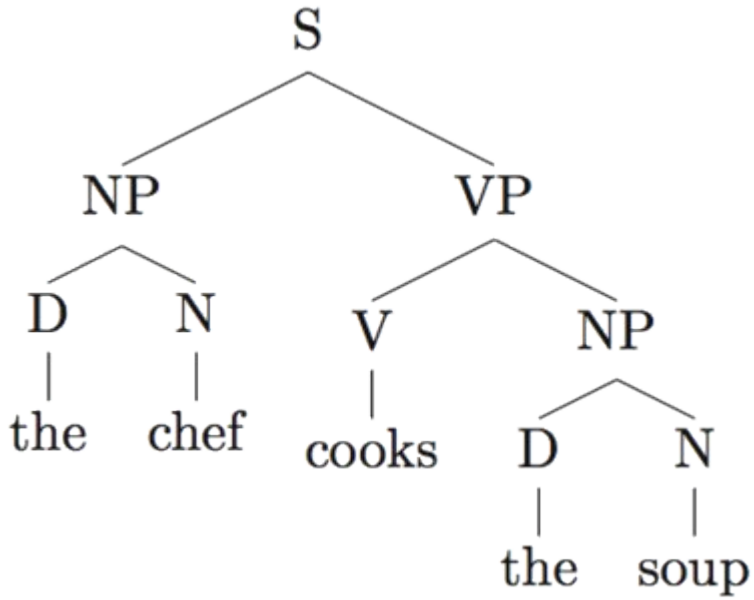
```
( (S
    (NP-SBJ (DT The) (NN move))
    (VP (VBD followed)
      (NP
        (NP (DT a) (NN round))
        (PP (IN of)
          (NP
            (NP (JJ similar) (NNS increases))
            (PP (IN by)
              (NP (JJ other) (NNS lenders)))
            (PP (IN against)
              (NP (NNP Arizona) (JJ real) (NN estate) (NNS loans))))))
      (, ,)
      (S-ADV
        (NP-SBJ (-NONE- *))
        (VP (VBG reflecting)
          (NP
            (NP (DT a) (VBG continuing) (NN decline))
            (PP-LOC (IN in)
              (NP (DT that) (NN market)))))))
    (. .)))
```

Most well known part is the Wall Street Journal section of the Penn TreeBank.

1 M words from the 1987-1989 Wall Street Journal newspaper.

Constituency (phrase structure)

- Phrase structure organizes words into nested constituents
- How do we know what is a constituent?
- Distribution: a constituent behaves as a unit that can appear in different places:
  - John talked [to the children] [about drugs].
  - John talked [about drugs] [to the children].
  - *John talked drugs to the children about
- Substitution/expansion/pro-forms:
  - I sat [on the box/right of the box/there].
- Bracket notation of a tree (Lisp S-structure)



(S (NP (N Fed)) (VP (V raises) (NP (N interest) (N rates))) draw it as a tree

Is string α a constituent?

He talks [in class].

- Substitution test:
  - Can α be replaced by a single word?
  - He talks [there].
- Movement test:
  - Can α be moved around in the sentence?
  - [In class], he talks.
- Answer test:
  - Can α be the answer to a question?
  - Where does he talk? - [In class].

Constituents: Heads and dependents

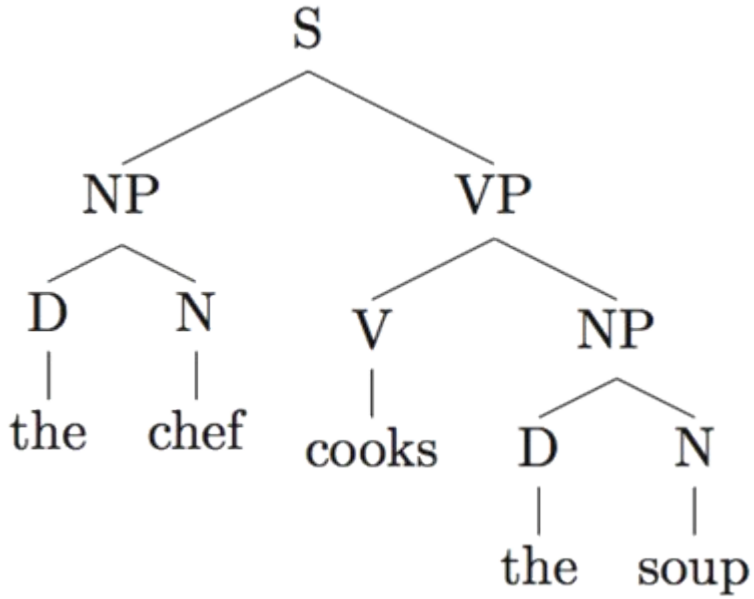
- There are different kinds of constituents
  - Noun phrases: the man, a girl with glasses, Iowa
  - Prepositional phrases: with glasses, in the garden
  - Verb phrases: eat sushi, sleep, sleep soundly
- Every phrase has a head:
  - Noun phrases: the man, a girl with glasses, Iowa
  - Prepositional phrases: with glasses, in the garden
  - Verb phrases: eat sushi, sleep, sleep soundly
  - The other parts are its dependents.
  - Dependents are either arguments or adjuncts

Slides adapted from https://courses.grainger.illinois.edu/cs447/fa2018/Slides/Lecture09.pdf

Context-free grammars

- A CFG is a 4-tuple consisting of:
  - A set of nonterminals (e.g. = {S, NP, VP, PP, Noun, Verb, ….})
  - A set of terminals (e.g. = {I, you, he, eat, drink, sushi, ball, })
  - A set of rules

  - A start symbol



Constituents: Heads and dependents

- There are different kinds of constituents
  - Noun phrases: the man, a girl with glasses, Iowa
  - Prepositional phrases: with glasses, in the garden
  - Verb phrases: eat sushi, sleep, sleep soundly
- Every phrase has a head:
  - Noun phrases: the man, a girl with glasses, Iowa
  - Prepositional phrases: with glasses, in the garden
  - Verb phrases: eat sushi, sleep, sleep soundly
  - The other parts are its dependents.
  - Dependents are either arguments or adjuncts

Slides adapted from https://courses.grainger.illinois.edu/cs447/fa2018/Slides/Lecture09.pdf

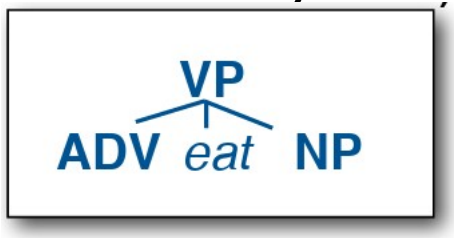
Heads, Arguments and Adjuncts in CFGs

- Heads: We assume that each RHS has one head, e.g.
  - VP → Verb NP (Verbs are heads of VPs)
  - NP → Det Noun (Nouns are heads of NPs)
  - S → NP VP (VPs are heads of sentences)
  - Exception: Coordination, lists: VP → VP conj VP
- Arguments: The head has a different category from the parent
  - VP → Verb NP (the NP is an argument of the verb)
- Adjuncts: The head has the same category as the parent
  - VP → VP PP (the PP is an adjunct)

Chomsky Normal Form

- The right-hand side of a standard CFG can have an arbitrary number of symbols (terminals and nonterminals):

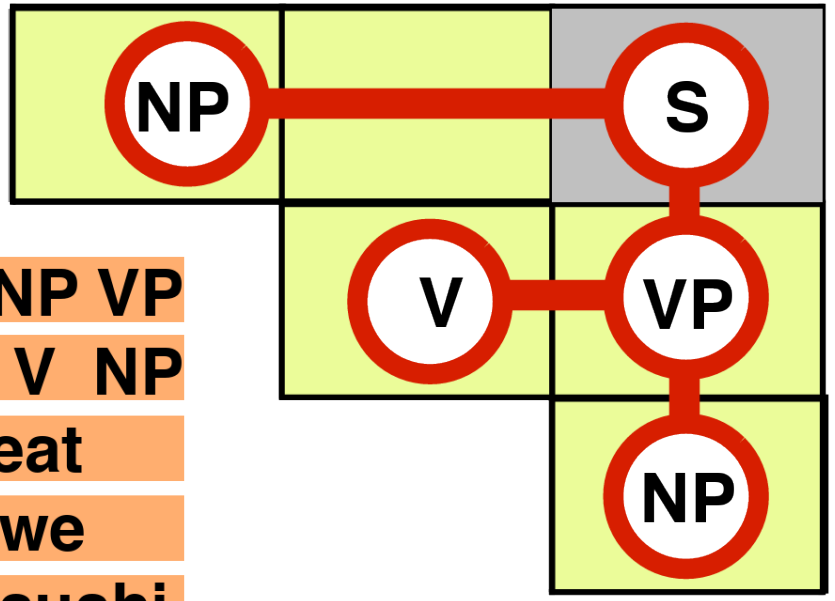
VP → ADV eat NP



- A CFG in Chomsky Normal Form (CNF) allows only two kinds of right-hand sides:
  - Two nonterminals: VP → ADV VP
  - One terminal: VP → eat
- Any CFG can be transformed into an equivalent CNF:

CKY chart parsing algorithm

- Bottom-up parsing:
  - start with the words
- Dynamic programming:
  - save the results in a table/chart
  - re-use these results in finding larger constituents
- Complexity:
  - : length of string
  - : size of grammar

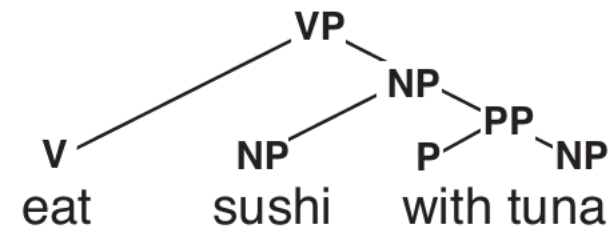


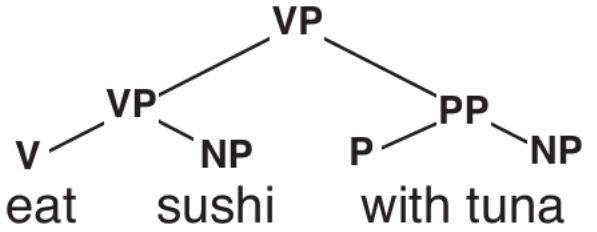
S → NP VP
VP → V NP
V → eat
NP → we
NP → sushi

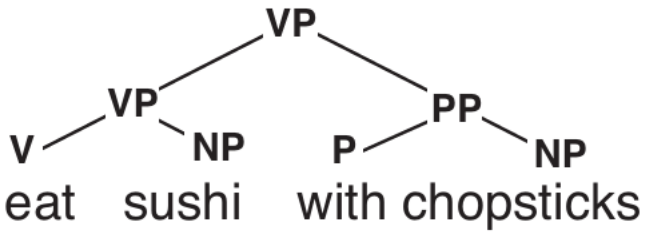
We eat sushi

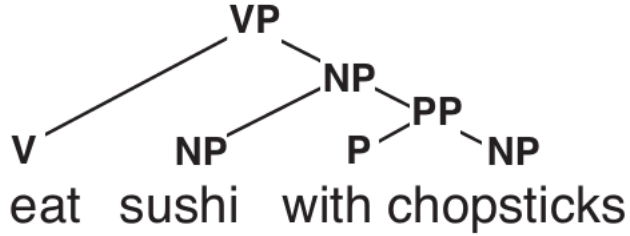
Ambiguity

- Eat sushi with tuna
- Eat sushi with chopsticks
- http://stanza.run/
- https://parser.kitaev.io/



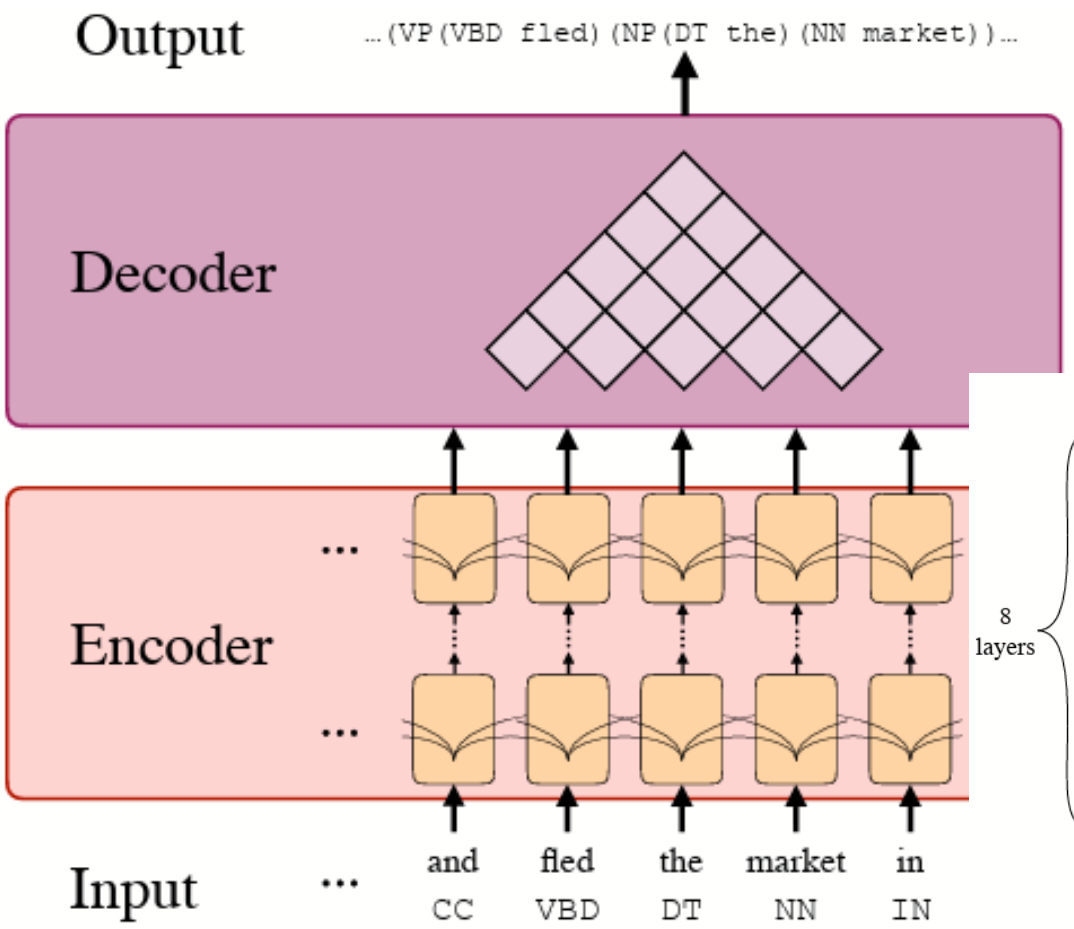


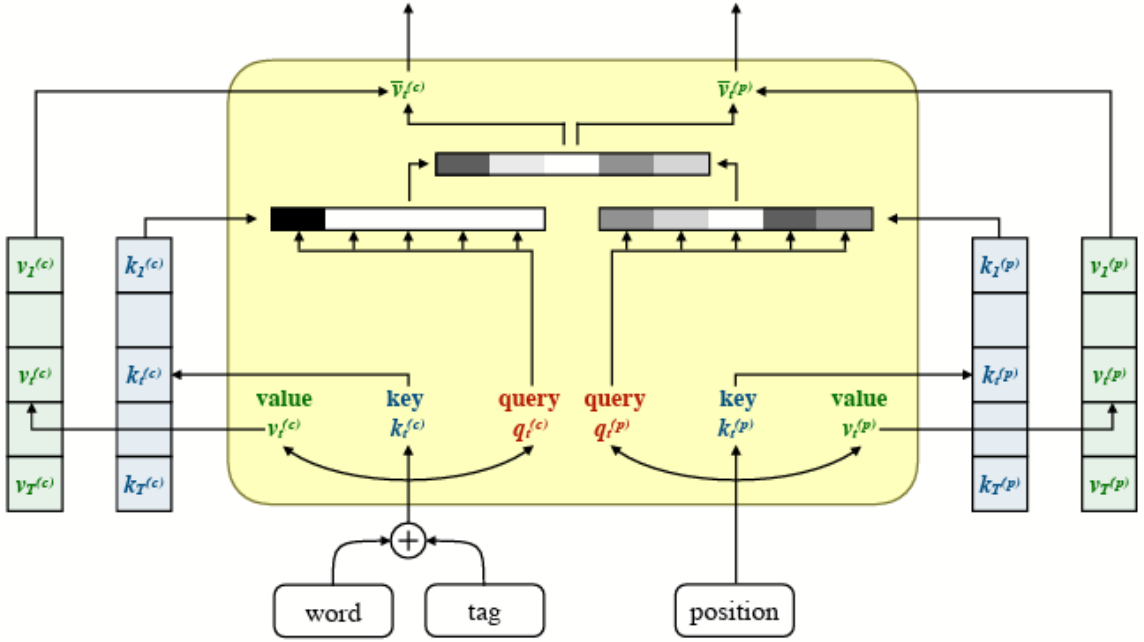


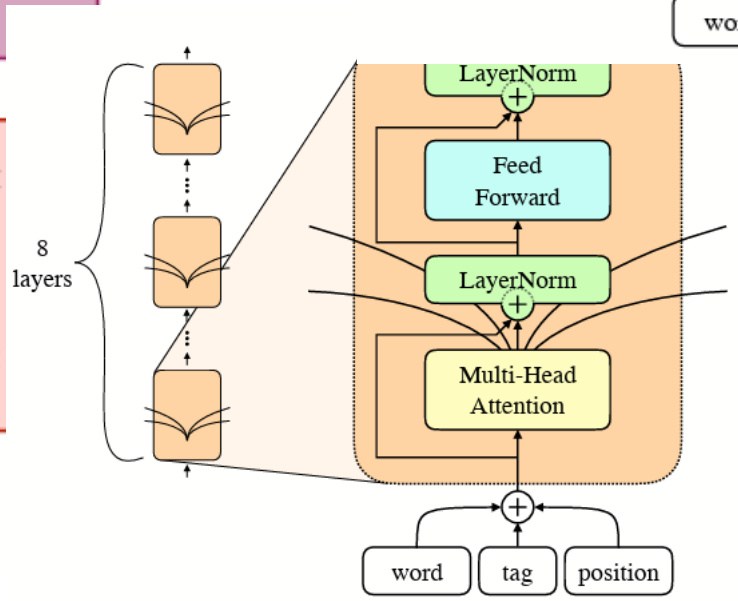


- What's the most likely parse for sentence S?
- We need a model of
- Idea: very similar to POS tagging

Constituency Parsing with a Self-Attentive Encoder







A single attention head, after factoring content and position information. Attention probabilities are calculated separately for the two types of information, and a combined probability distribution is then applied to both types of input information.

Span-based Constituency Parsing

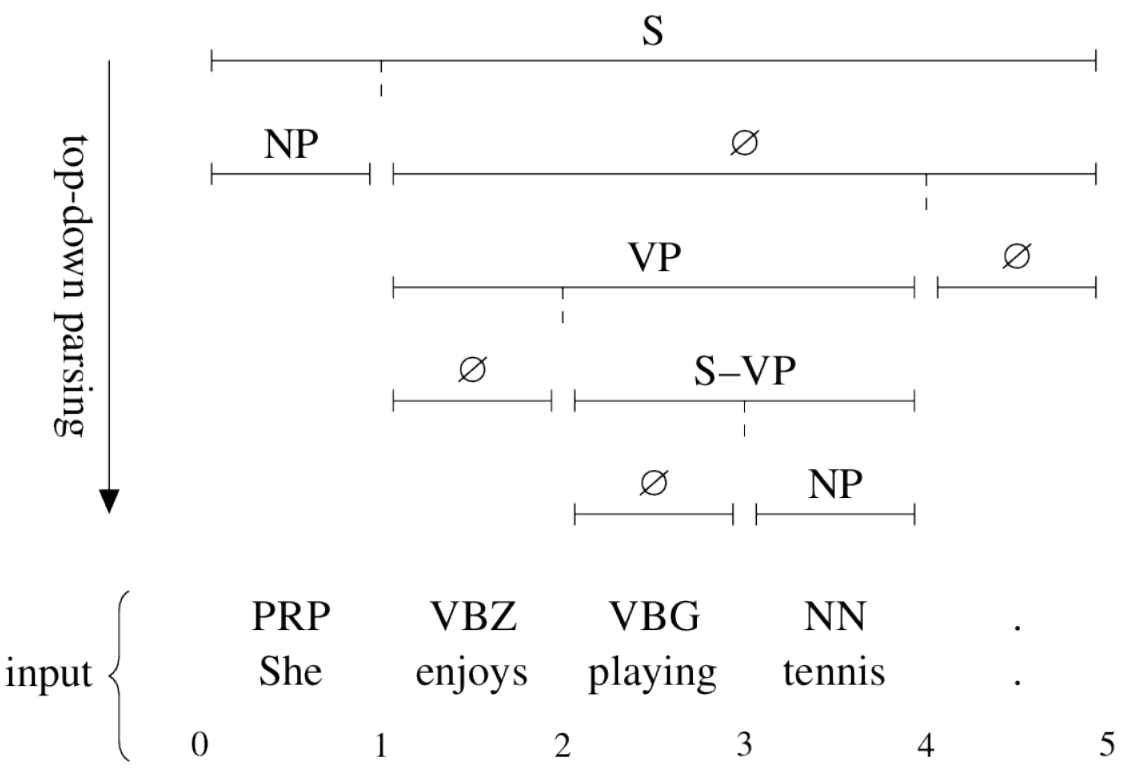
- A constituency tree is represented as a set of labeled spans
  - begins at position
  - ends at position
  - has label
- The parser assigns a score to each tree
  - are produced by a neural network
- Find the highest scoring valid tree
  - Achieved by a modified CKY recursion
  - running time:

A Minimal Span-Based Neural Constituency Parser

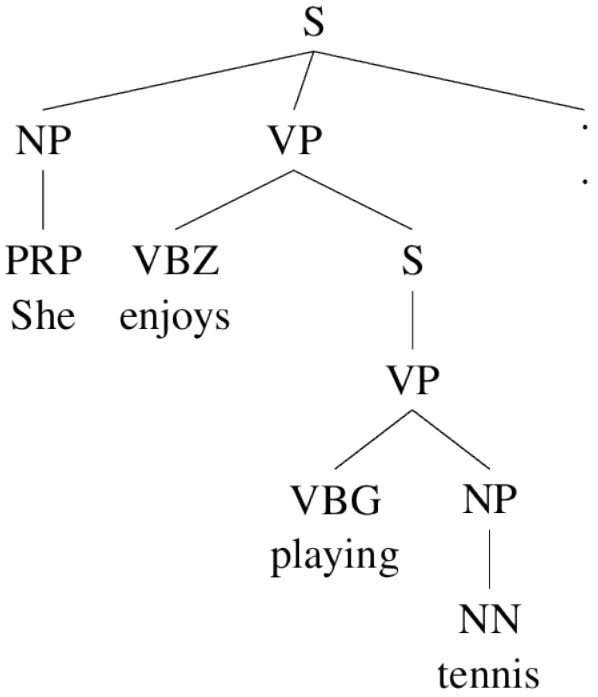
. Mitchell Stern, Jacob Andreas, Dan Klein. ACL 2017.

Top-Down Parsing

- given a span
  - assign it a label
  - pick a split point
  - Repeat
  - running time:



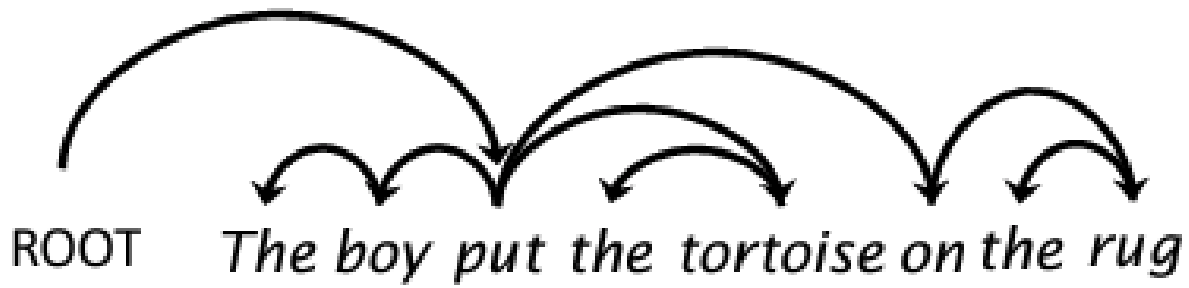
(a) Execution of the top-down parsing algorithm.

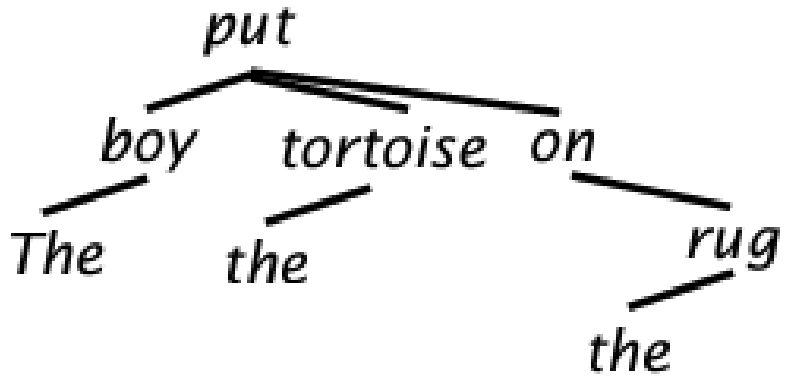


(b) Output parse tree.

Dependency structure

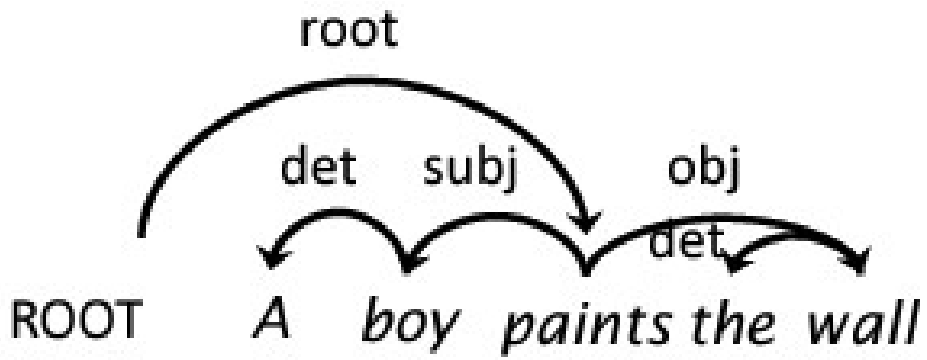
- In a dependency grammar framework, a parse is a tree where
  - the nodes stand for the words in an utterance
  - The links between the words represent dependency relations between pairs of words.
    - Relations may be typed (labeled), or not.





Dependency Labels

- Argument dependencies:
  Subject (subj), object (obj), indirect object (iobj)…
- Modifier dependencies:
  Determiner (det), noun modifier (nmod), verbal modifier (vmod), etc.



Comparison

- Dependency structures explicitly represent
  - head-dependent relations (directed arcs)
  - functional categories (arc labels)
- Phrase structures explicitly represent
  - phrases (nonterminal nodes),
  - structural categories (nonterminal labels),
  - possibly some functional categories (grammatical functions, e.g. PP-LOC)
- There exist also hybrid approaches, e.g. Dutch Alpino grammar.

Two approaches to dependency parsing

- Transition-based parsing
  - Proceed through a sequence of actions, building up a representation step by step
  - The representation, and any step, depends on the representations that came before
- Graph-based parsing
  - Start with probabilities for each edge
  - Apply some sort of dynamic programming

slides adopted from https://michaelmilleryoder.github.io/cs2731_spring2024/slides/session21_cs2731_spring2024.pdf

Transition-based parsing

- Process input from left-to-right once, making a sequence of greedy parsing decisions
- Represents the current state/configuration of the parse:
  - Stack
  - Buffer
  - Current set of relations
- In arc-standard parsing, possible actions are:
  - SHIFT: move first word in the buffer to the stack
  - LEFT-ARC: draw an arc from word in the top of the stack to second word in the stack; remove dependent word (second word)
  - RIGHT-ARC: draw an arc from second word in the stack to the top of the stack; remove dependent word (top of the stack)

Book me the morning flight

Step	Stack	Word list	Action	Relation added
0	[root]	[book, me, the, morning, flight]	SHIFT	

Book me the morning flight

Step	Stack	Word list	Action	Relation added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	

Book me the morning flight

Step	Stack	Word list	Action	Relation added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	

Book me the morning flight

Step	Stack	Word list	Action	Relation added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book -> me)

Book me the morning flight

Step	Stack	Word list	Action	Relation added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book -> me)
3	[root, book]	[the, morning, flight]	SHIFT	

Book me the morning flight

Step	Stack	Word list	Action	Relation added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book -> me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	

Book me the morning flight

Step	Stack	Word list	Action	Relation added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book -> me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	

Book me the morning flight

Step	Stack	Word list	Action	Relation added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book -> me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	

Book me the morning flight

Step	Stack	Word list	Action	Relation added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book -> me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning <- flight)

Book me the morning flight

Step	Stack	Word list	Action	Relation added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book -> me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning <- flight)
7	[root, book, the, flight]	[]	LEFTARC	

Book me the morning flight

Step	Stack	Word list	Action	Relation added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book -> me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning <- flight)
7	[root, book, the, flight]	[]	LEFTARC	(the <- flight)

Book me the morning flight

Step	Stack	Word list	Action	Relation added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book -> me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning <- flight)
7	[root, book, the, flight]	[]	LEFTARC	(the <- flight)
8	[root, book, flight]	[]	RIGHTARC	

Book me the morning flight

Step	Stack	Word list	Action	Relation added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book -> me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning <- flight)
7	[root, book, the, flight]	[]	LEFTARC	(the <- flight)
8	[root, book, flight]	[]	RIGHTARC	(book -> flight)

Book me the morning flight

Step	Stack	Word list	Action	Relation added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book -> me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning <- flight)
7	[root, book, the, flight]	[]	LEFTARC	(the <- flight)
8	[root, book, flight]	[]	RIGHTARC	(book -> flight)
9	[root, book]	[]	RIGHTARC	

Book me the morning flight

Step	Stack	Word list	Action	Relation added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book -> me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning <- flight)
7	[root, book, the, flight]	[]	LEFTARC	(the <- flight)
8	[root, book, flight]	[]	RIGHTARC	(book -> flight)
9	[root, book]	[]	RIGHTARC	(root -> book)

Book me the morning flight

Step	Stack	Word list	Action	Relation added
0	[root]	[book, me, the, morning, flight]	SHIFT	
1	[root, book]	[me, the, morning, flight]	SHIFT	
2	[root, book, me]	[the, morning, flight]	RIGHTARC	(book -> me)
3	[root, book]	[the, morning, flight]	SHIFT	
4	[root, book, the]	[morning, flight]	SHIFT	
5	[root, book, the, morning]	[flight]	SHIFT	
6	[root, book, the, morning, flight]	[]	LEFTARC	(morning <- flight)
7	[root, book, the, flight]	[]	LEFTARC	(the <- flight)
8	[root, book, flight]	[]	RIGHTARC	(book -> flight)
9	[root, book]	[]	RIGHTARC	(root -> book)
10	[root]	[]	DONE	

The Core of Transition-based Parsing

- At each iteration, choose among {SHIFT, RIGHT-ARC, LEFT-ARC}.
- Training data: Dependency treebank trees converted into "oracle" transition sequence.
  - These transition sequences give the right tree,
  - pairs: <state, correct_transition>.
  - Each word gets SHIFTed **once** and participates as a child in **one** ARC.

Features for Transition Parsing Come from the Configuration

- Where do the features for making parsing decisions come from?
  - The words in the buffer
  - The words in the stack (e.g. the roots of the trees)
  - The children of these roots
  - The POS tags of the words
  - History of actions
- Feature combinations are important:
  - When parsing English, suppose that the second word in Sis a verb and the first is a noun.
  - The model should probably choose LEFT-ARC

Transition Parser

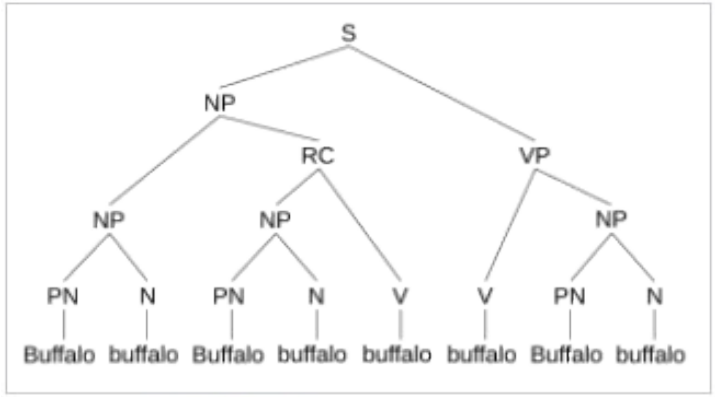
- Input: sentence
- Output: a sequence of operations
- Sequence-to-sequence model (S2S)
- **Potential flaw:** the classifier is typically trained under the assumption that previous classification decisions were all correct. As yet, there is no principled solution to this problem, but there are approximations based on dynamic oracles".

Graph-based Parser

- assign a weight to each possible edge
- construct a tree from these weighted edges
  - maximum spanning tree (MST)

Relevance to other NLP Tasks

- Machine Translation
  - Better sentence alignment and structure preservation.
- QA Systems
  - Understand question intent and locate relevant answers.
- Semantic Role Labeling
- Information Extraction



Simplified parse tree:
S = sentence
NP = noun phrase
RC = relative clause
VP = verb phrase
PN = proper noun
N = noun
V = verb

Semantic Role Labeling

- Assigns roles to words or phrases in a sentence.
- Identifies the predicate (action) and its arguments (who did what to whom, when, where, etc.)

Sentence: 'John gave Mary a book.'
SRL Output:
- Predicate: gave
- Agent (Who): John
- Recipient (To Whom): Mary
- Theme (What): a book

Sentence: 'John gave a book to Mary.'
SRL Output:
- Predicate: gave
- Agent (Who): John
- Recipient (To Whom): Mary
- Theme (What): a book

Sentence Simplification

- E.g. *The subcutaneous injection of isoproterenol (30 mg/kg) into rats twice at an interval of 24 h, for two consecutive days, significantly increased serum lactate dehydrogenase, creatine phosphokinase, alanine transaminase, aspartate transaminase, and angiotensin-converting enzyme activities, total cholesterol, triglycerides, free serum fatty acid, cardiac tissue malondialdehyde (MDA), and nitric oxide levels and significantly decreased levels of glutathione and superoxide dismutase in cardiac tissue as compared to the normal control group ($P < 0.05$).*

- E.g. *The subcutaneous* ***injection*** *of isoproterenol (30 mg/kg) into rats twice at an interval of 24 h, for two consecutive days,* ***significantly increased*** *serum lactate dehydrogenase, creatine phosphokinase, alanine transaminase, aspartate transaminase, and angiotensin-converting enzyme activities, total cholesterol, triglycerides, free serum fatty acid, cardiac tissue malondialdehyde (MDA), and nitric oxide levels* ***and significantly decreased*** *levels of glutathione and superoxide dismutase in cardiac tissue as compared to the normal control group (P < 0.05).*

- E.g. *The subcutaneous* ***injection*** *of isoproterenol (30 mg/kg) into rats twice at an interval of 24 h, for two consecutive days,* ***significantly increased*** *serum lactate dehydrogenase, creatine phosphokinase, alanine transaminase, aspartate transaminase, and angiotensin-converting enzyme activities, total cholesterol, triglycerides, free serum fatty acid, cardiac tissue malondialdehyde (MDA), and nitric oxide levels* ***and significantly decreased*** *levels of glutathione and superoxide dismutase in cardiac tissue as compared to the normal control group (P < 0.05).*

Open Information Extraction (OpenIE)

The U.S. president Barack Obama gave his speech on Tuesday to thousands of people.

->

(Barack Obama, is the president of, the U.S.)

(Barack Obama, gave, his speech)

(Barack Obama, gave his speech, on Tuesday)

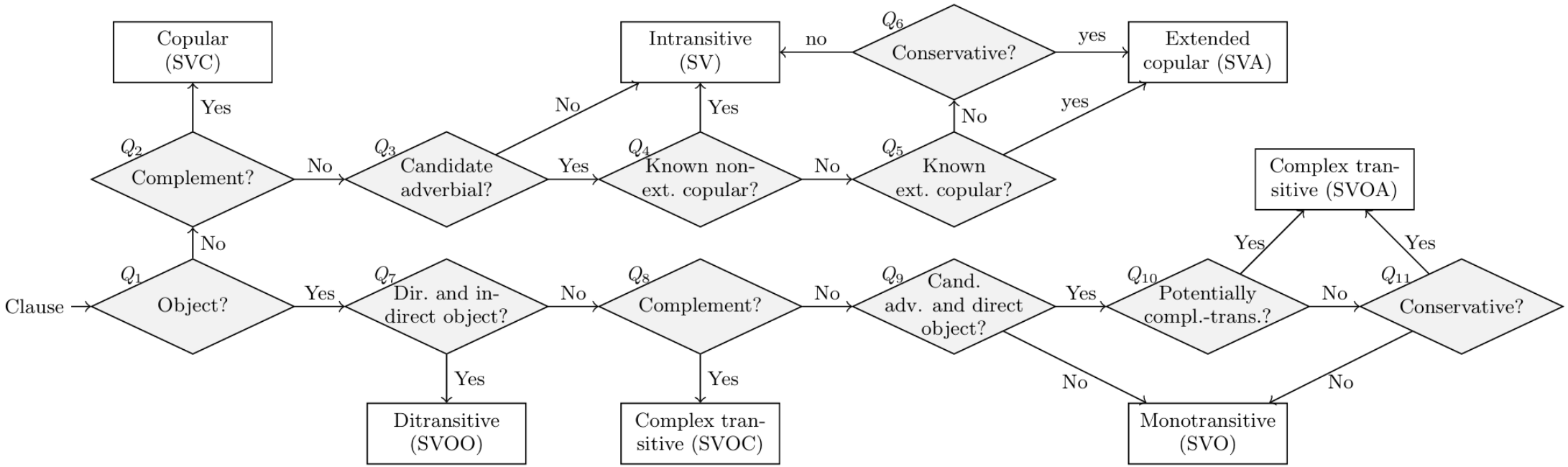
(Barack Obama, gave his speech, to thousands of people)

Grammar based OpenIE (ClausIE)

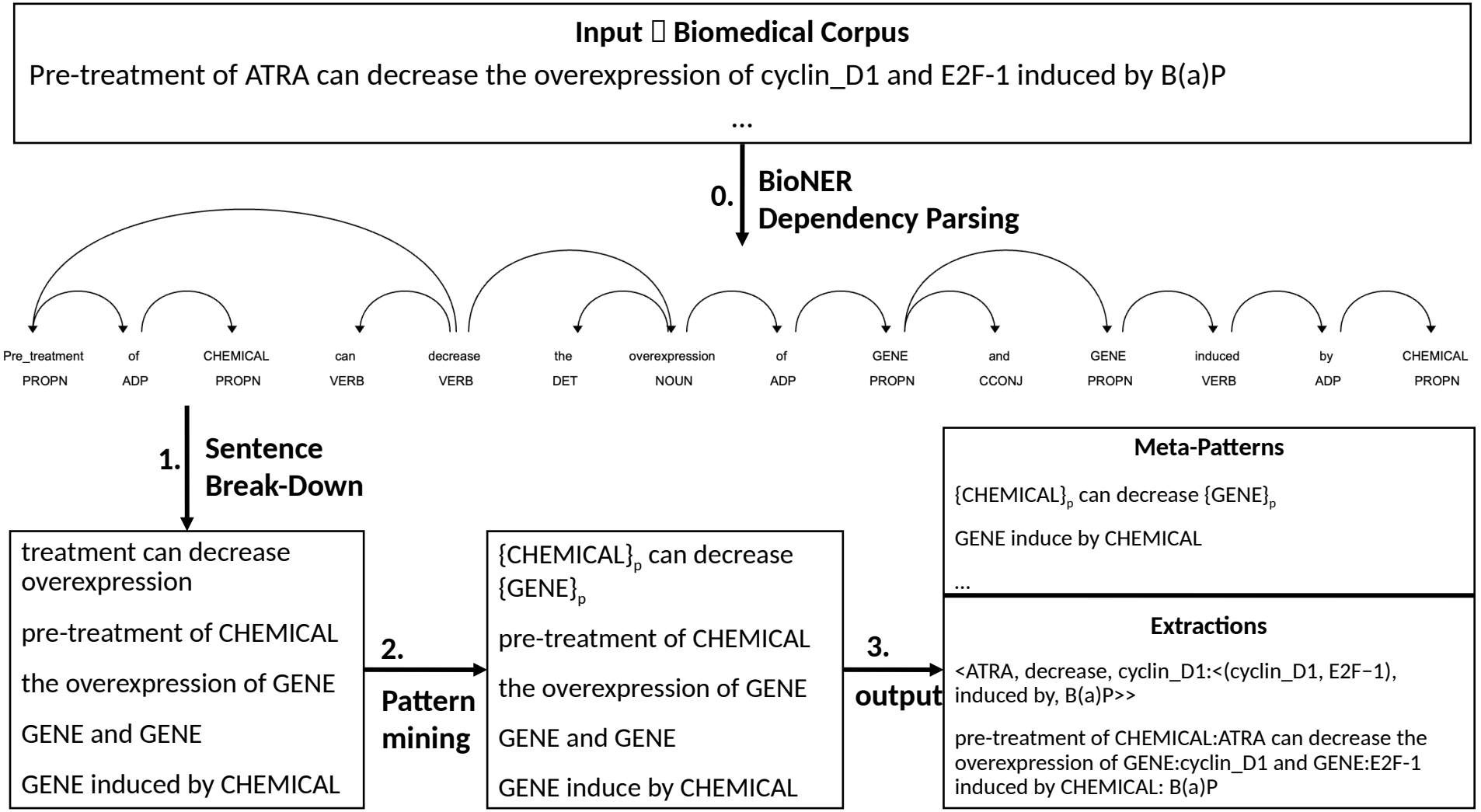
	Pattern	Clause type	Example	Derived clauses
			Basic patterns	
S_1:	SV_i	SV	AE died.	(AE, died)
S_2:	SV_eA	SVA	AE remained in Princeton.	(AE, remained, in Princeton)
S_3:	SV_cC	SVC	AE is smart.	(AE, is, smart)
S_4:	$SV_{mt}O$	SVO	AE has won the Nobel Prize.	(AE, has won, the Nobel Prize)
S_5:	$SV_{dt}O_iO$	SVOO	RSAS gave AE the Nobel Prize.	(RSAS, gave, AE, the Nobel Prize)
S_6:	$SV_{ct}OA$	SVOA	The doorman showed AE to his office.	(The doorman, showed, AE, to his office)
S_7:	$SV_{ct}OC$	SVOC	AE declared the meeting open.	(AE, declared, the meeting, open)
			Some extended patterns	
S_8:	SV_iAA	SV	AE died in Princeton in 1955.	(AE, died) (AE, died, in Princeton) (AE, died, in 1955) (AE, died, in Princeton, in 1955)
S_9:	SV_eAA	SVA	AE remained in Princeton until his death.	(AE, remained, in Princeton) (AE, remained, in Princeton, until his death)
S_{10}:	SV_cCA	SVC	AE is a scientist of the 20th century.	(AE, is, a scientist) (AE, is, a scientist, of the 20th century)
S_{11}:	$SV_{mt}OA$	SVO	AE has won the Nobel Prize in 1921.	(AE, has won, the Nobel Prize) (AE, has won, the Nobel Prize, in 1921)
S_{12}:	$ASV_{mt}O$	SVO	In 1921, AE has won the Nobel Prize.	(AE, has won, the Nobel Prize) (AE, has won, the Nobel Prize, in 1921)

S: Subject, V: Verb, C: Complement, O: Direct object, O_i: Indirect object, A: Adverbial, V_i: Intransitive verb, V_c: Copular verb, V_c: Extended-copular verb, V_{mt}: Monotransitive verb, V_{dt}: Ditransitive verb, V_{ct}: Complex-transitive verb

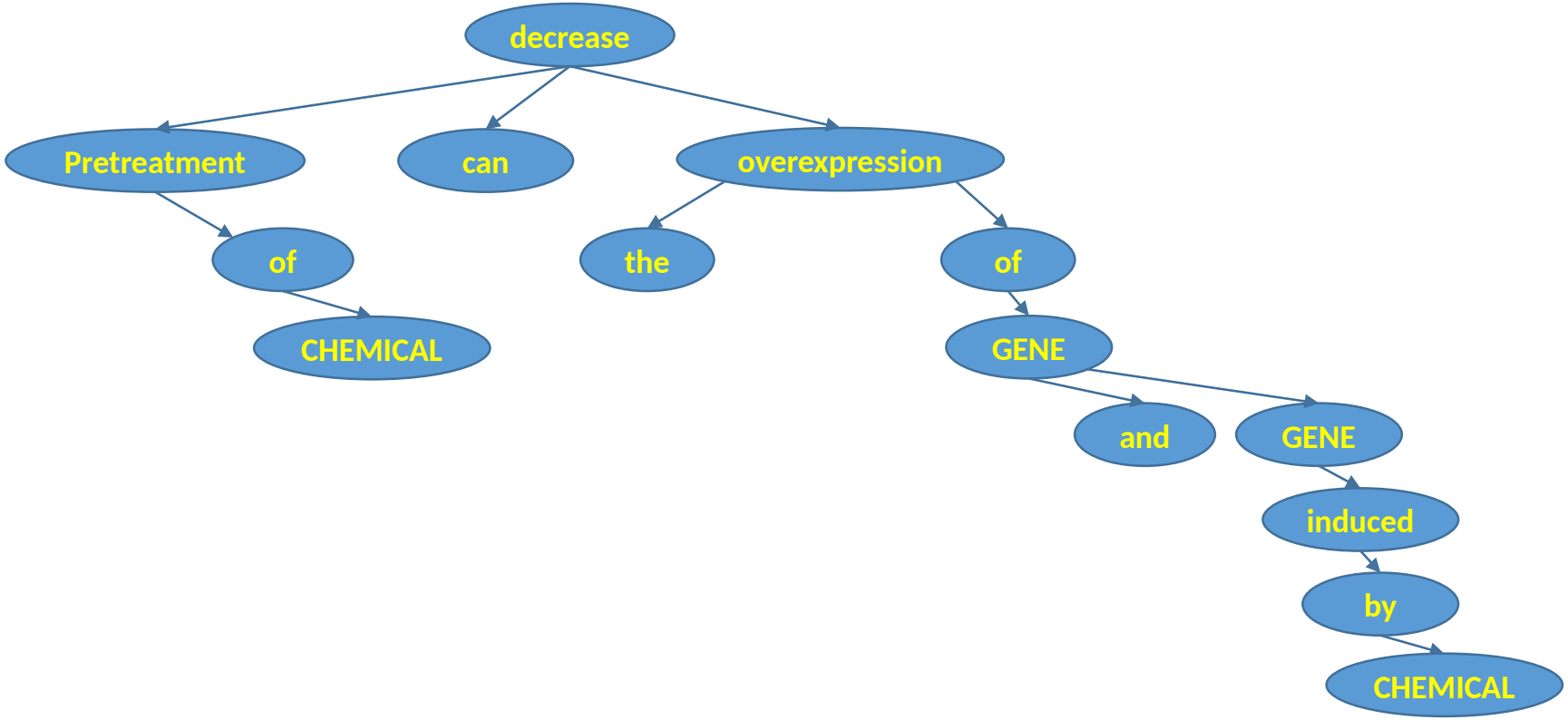
Work Flow


Clause
Q_1
Object?
Yes
Q_7
Dir. and in-direct object?
No
Q_8
Complement?
No
Q_9
Cand. adv. and direct object?
Yes
Q_{10}
Potentially compl.-trans.?
No
Q_{11}
Conservative?
No
Yes
Ditransitive (SVOO)
Yes
Complex tran-sitive (SVOC)
No
Monotransitive (SVO)
Yes
Complex tran-sitive (SVOA)
Yes
Q_2
Complement?
Yes
Copular (SVC)
No
Q_3
Candidate adverbial?
No
Intransitive (SV)
Yes
Q_4
Known non-ext. copular?
No
Q_5
Known ext. copular?
yes
Extended copular (SVA)
No
Q_6
Conservative?
no
Intransitive (SV)
yes
Extended copular (SVA)

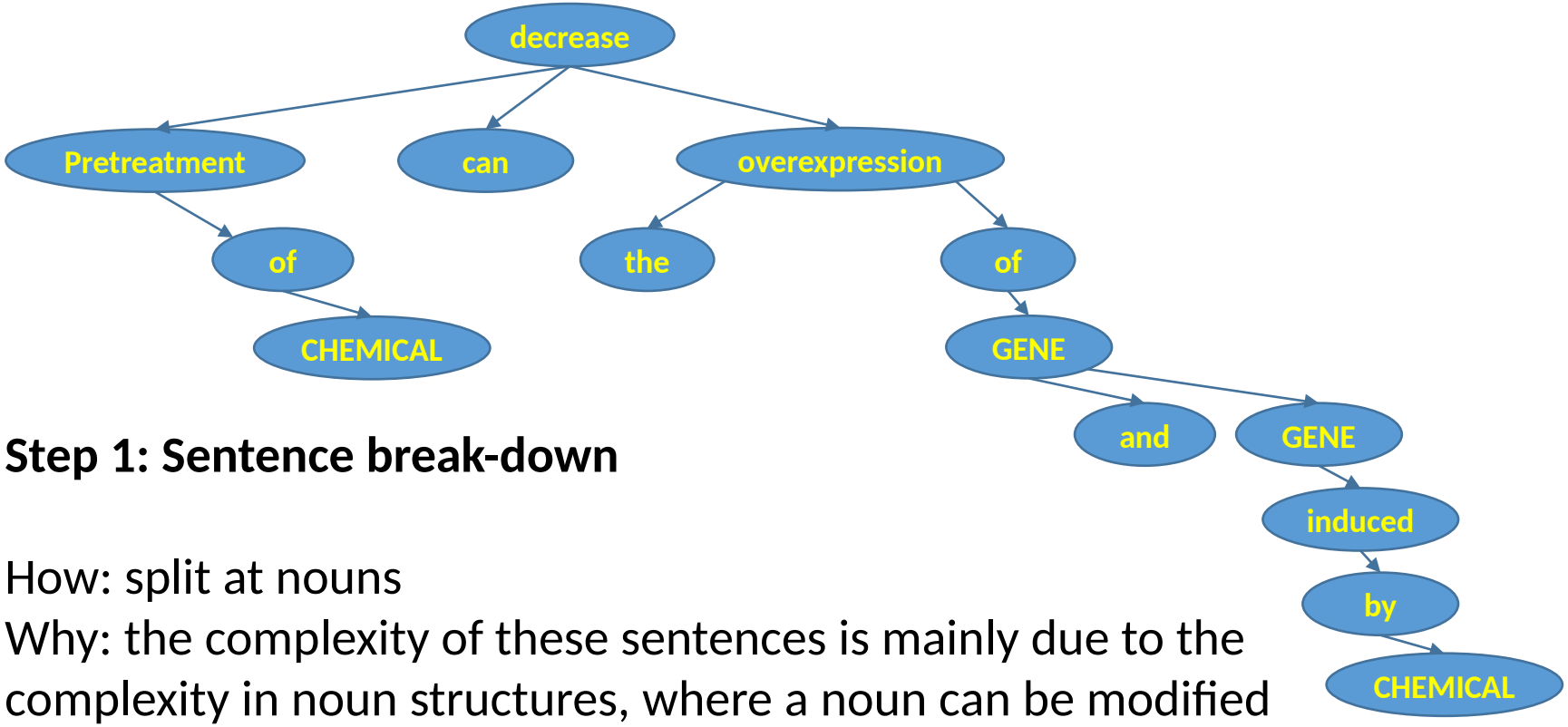
Pattern Based Relation Extraction



Pre-treatment of [ATRA]$_{CHEMICAL}$ can decrease the overexpression of [cyclin_D1]$_{GENE}$ and [E2F-1]$_{GENE}$ induced by [B(a)P]$_{CHEMICAL}$.

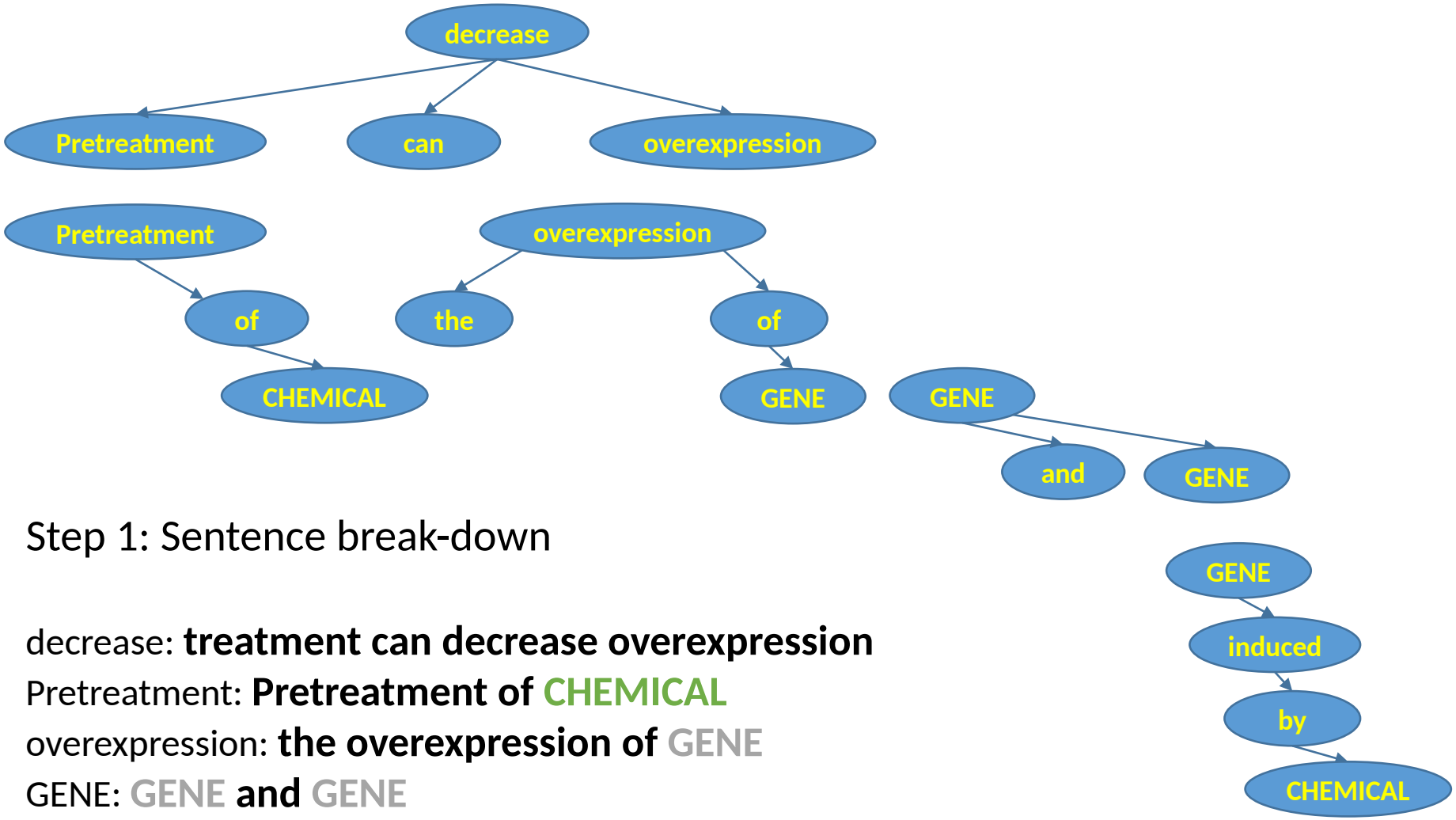


Pre-treatment of [ATRA]$_{CHEMICAL}$ can decrease the overexpression of [cyclin_D1]$_{GENE}$ and [E2F-1]$_{GENE}$ induced by [B(a)P]$_{CHEMICAL}$.



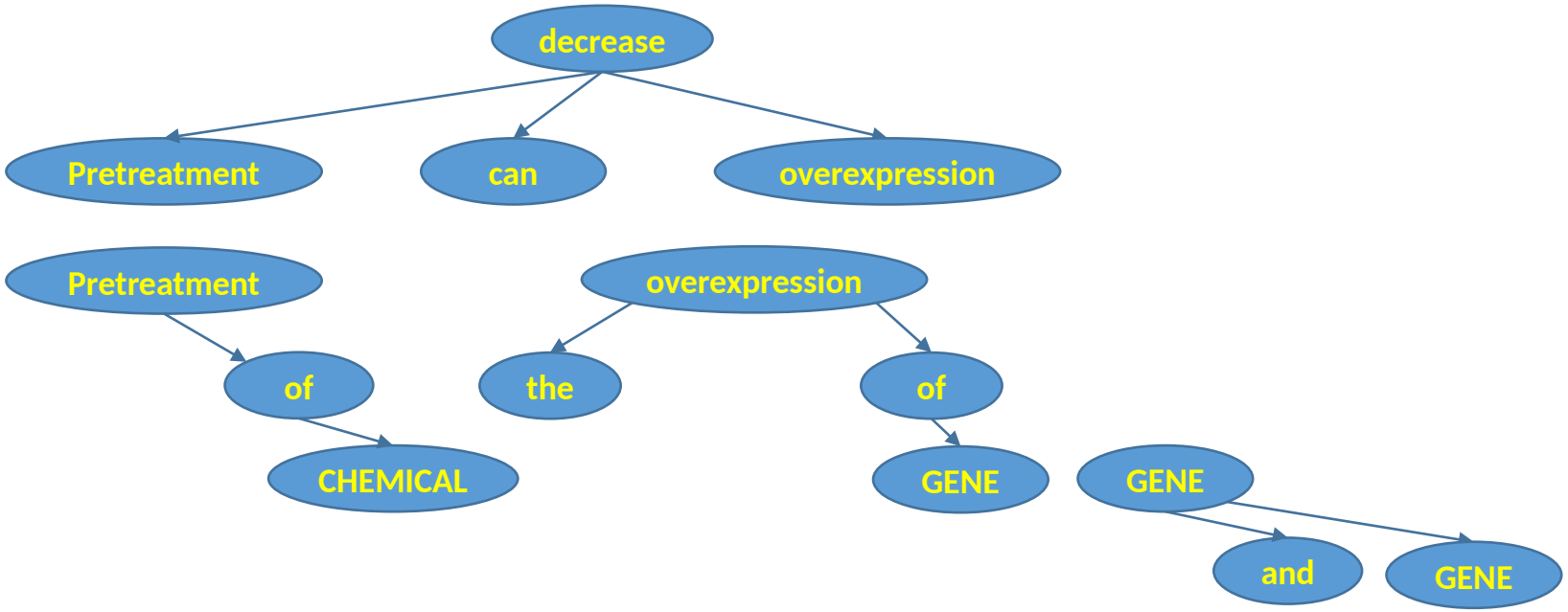
Step 1: Sentence break-down

How: split at nouns
Why: the complexity of these sentences is mainly due to the complexity in noun structures, where a noun can be modified by other nouns, adjectives, adjectival clauses, etc.



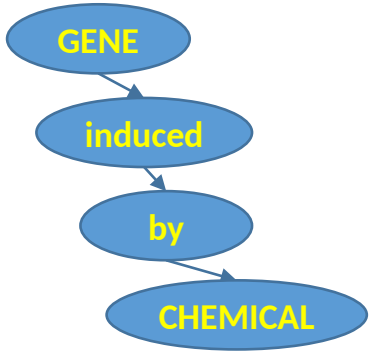
Step 1: Sentence break-down

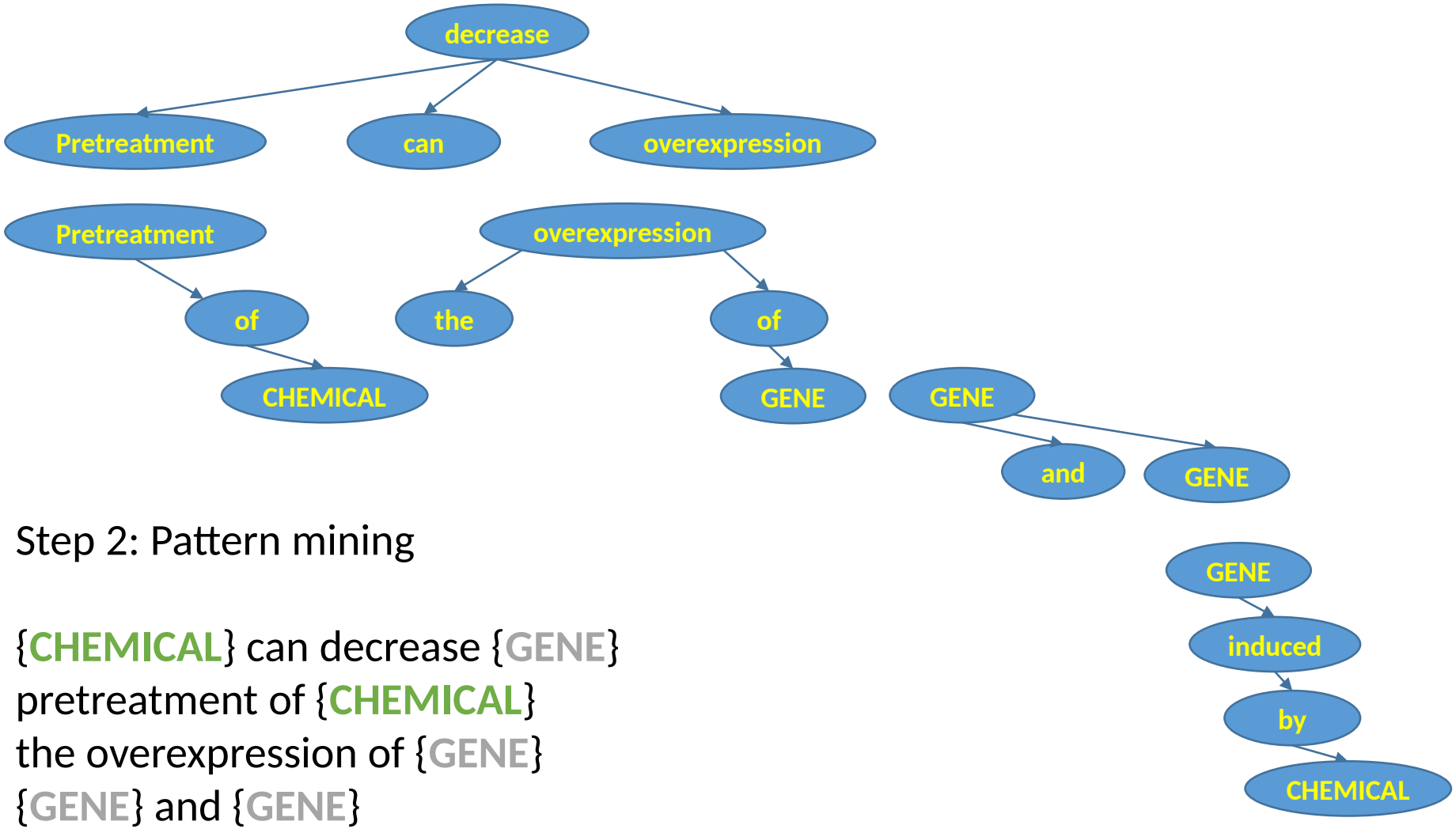
decrease: **treatment can decrease overexpression**
Pretreatment: **Pretreatment of** CHEMICAL
overexpression: **the overexpression of** GENE
GENE: GENE **and** GENE
GENE: GENE **induced by** CHEMICAL



Step 2: Pattern mining on short sentences

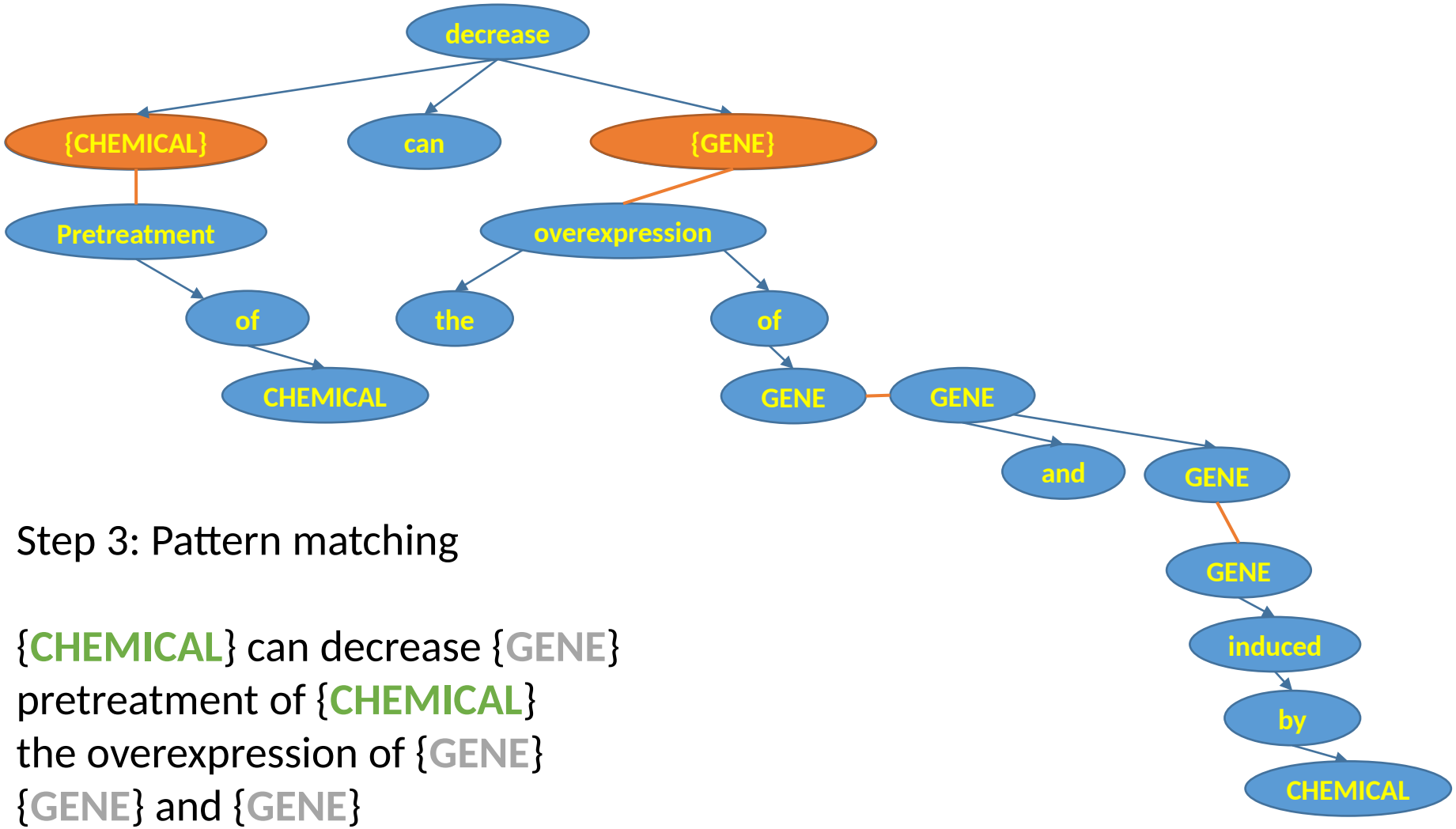
- **Constraints to ensure the patterns have complete semantic meanings**





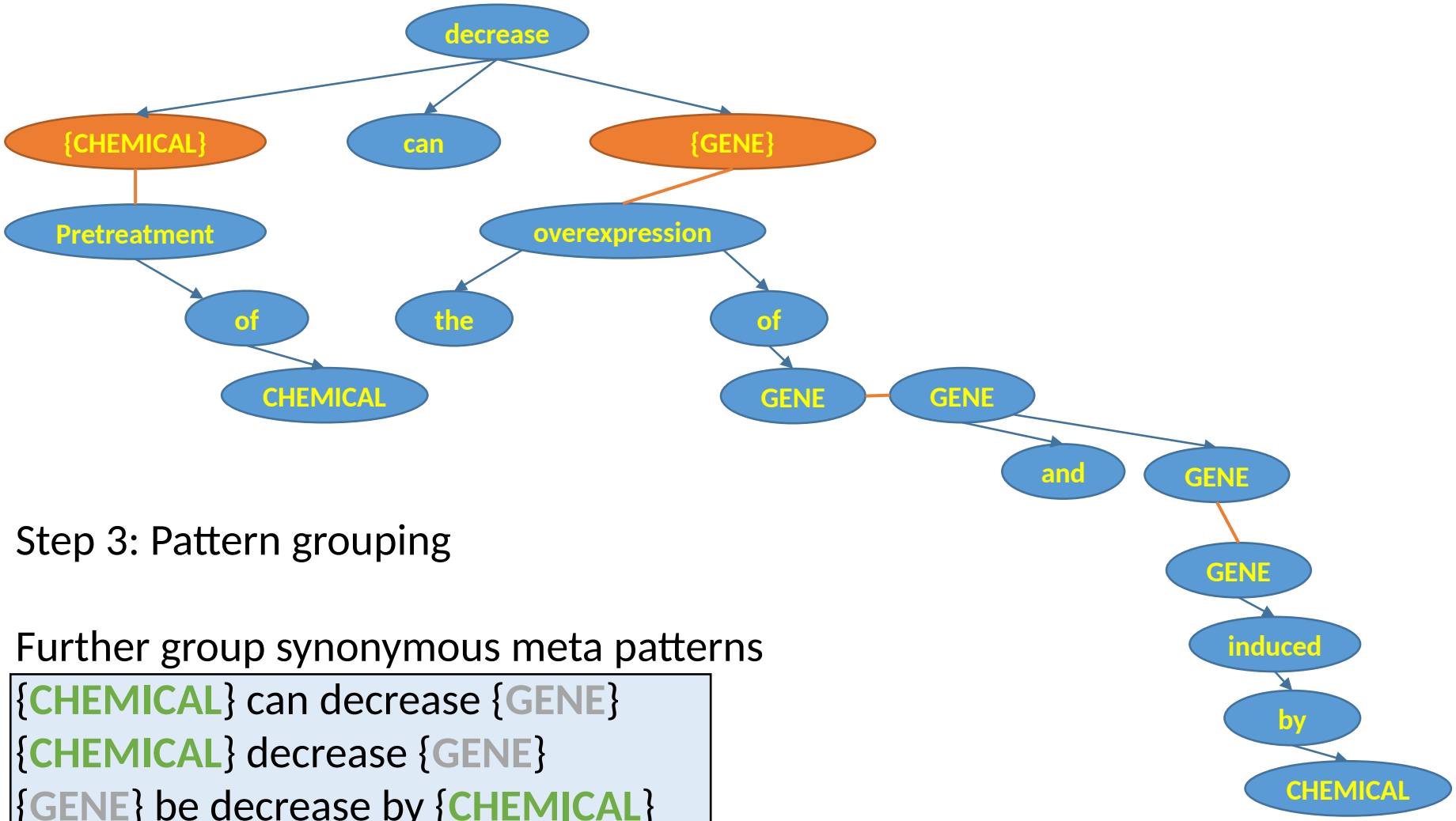
Step 2: Pattern mining

{CHEMICAL} can decrease {GENE}
pretreatment of {CHEMICAL}
the overexpression of {GENE}
{GENE} and {GENE}
{GENE} induced by {CHEMICAL}



Step 3: Pattern matching

{CHEMICAL} can decrease {GENE}
pretreatment of {CHEMICAL}
the overexpression of {GENE}
{GENE} and {GENE}
{GENE} induced by {CHEMICAL}



Step 3: Pattern grouping

Further group synonymous meta patterns

{CHEMICAL} can decrease {GENE}
{CHEMICAL} decrease {GENE}
{GENE} be decrease by {CHEMICAL}