

DEEP PARSING IN WATSON

Presented by
Jeffrey Kabot

DEEP PARSING

- Consists of:
 - English Slot Grammar (ESG) parser
 - Predicate-Argument Structure (PAS) builder
- Provides an analytical structure of questions posed and textual knowledge.

DEEP PARSING

■ English Slot Grammar

- Deep parser which explores the syntactic and logical structure of a sentence.
- Seeks to generate semantic clues based on a network of syntactic relationships.

■ Predicate-Argument Structure

- Builder which reduces the complexity of results given by the ESG.
- Result is more general form which is logically approximate to the result of the original parse.
- E.g. “John sold a fish” and “A fish was sold by John” yield different parse trees via ESG but reduce to the same PAS.
- Seeks to expand the space of textual evidence which would match well to the question.

PURPOSE

- **Relation Extraction** – identify semantic relationships among entities
- **Question Analysis** – identify type of answer sought
 - e.g. person, place
- **Keyword Search** – seeks entities with strong semantic relation to question's answer
- **Passage-Scoring** – determine degree of a passage's alignment to question

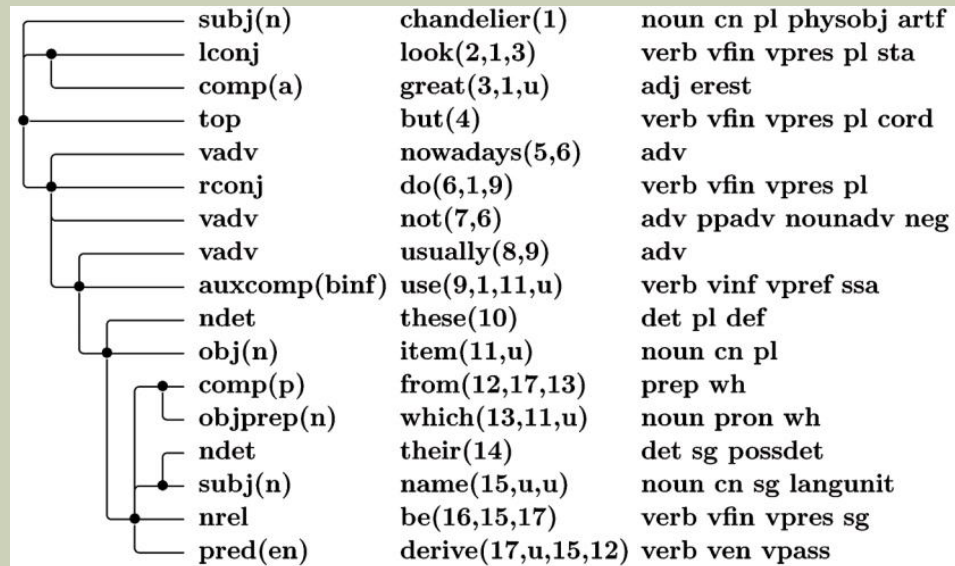
SLOT GRAMMAR PARSING

- **Steps:**

- 1) Tokenization
- 2) Morpholexical Analysis
- 3) Syntactic Analysis

PARSE TREES

- Consists of tree nodes each centered on a headword
- Each node N is modified by left and right modifiers M
- Each modifier M is a tree node of its own
 - We say that a modifier M fills a slot in N
 - The slot is the grammatical role of M in N
 - This parse tree defines the surface structure of a sentence.



SLOTS

- Complement slots – determined by the properties of their headword (e.g. verb which take subjects and objects)

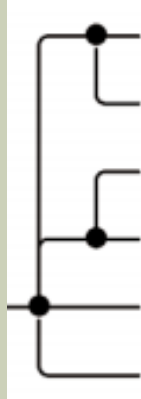
subj(n)	chandelier(1)	noun cn pl physobj artf
lconj	look(2,1,3)	verb vfin vpres pl sta
comp(a)	great(3,1,u)	adj erest

- Adjunct slots – determined by the part of speech of their headword (e.g. verbs can take an adverb)

vadv	usually(8,9)	adv
auxcomp(bin)	use(9,1,11,u)	verb vinf vpref ssa

PARSE NODES

- Consist of:
 - Headword
 - ID
 - Logical argument frame
 - Features
 - Modifier structure

	comp(p)	from(12,17,13)	prep wh
	objprep(n)	which(13,11,u)	noun pron wh
	ndet	their(14)	det sg possdet
	subj(n)	name(15,u,u)	noun cn sg langunit
	nrel	be(16,15,17)	verb vfin vpres sg
	pred(en)	derive(17,u,15,12)	verb ven vpass

DEEP PARSING

- Surface structure is determined by modifier structure
- Deep structure is determined by logical argument frames

comp(p)	from(12,17,13)	prep wh
objprep(n)	which(13,11,u)	noun pron wh
ndet	their(14)	det sg possdet
subj(n)	name(15,u,u)	noun cn sg langunit
nrel	be(16,15,17)	verb vfin vpres sg
pred(en)	derive(17,u,15,12)	verb ven vpass

SG LEXICONS

■ Morpholexical Analysis: how to determine frames?

- Look up word in the provided SG Lexicon and match its use in context to a sense frame specified in the Lexicon

```
talk < v (obj n (p about)) (comp (p to with))  
      < v obj1 (comp1 (p into))  
      < n nsubj (nobj n (p about))  
          (ncomp (p to with))
```

■ Lexical entries

■ Consist of:

- Part of speech – e.g. noun, verb, adjective, etc.
- Complement slot frame
- Features – syntactic features or semantic types e.g. object, property, event, living being
- Numerical score – rate sense frames
- Subject area – e.g. computers, medicine
- Support verb construction

IBM IMPROVEMENTS ON SG LEXICON

- Match noun frames with verb frames
 - E.g. encode a relationship between “celebration and “celebrate”
 - Helps match questions to answers
- Augmentation of ESG base lexicon (using WordNet)
 - Increases number of entries
 - Indicates semantic types
- Noun-verb correspondances
 - E.g. verb defer has indicated noun-forms deferral, deference, etc.
- Chunk Lexicons
 - Handle multiword entries (e.g. “Sing a Song of Sixpence”)
- LAT Reward Features
 - Aid in identification of answer types

SYNTACTIC ANALYSIS

- Combine tokens into syntactic constituents
- Bottom-up, left-right organization of constituents into slots
- Subtrees build phrases
- Phrases are scored according to lexical use of constituents, rules in grammar

PREDICATE-ARGUMENT STRUCTURE BUILDER

- Simplifies and generalizes result of ESG parse
 - Elements change exact semantic meaning but in general are not essential to its core meaning.
 - Does not process original text. Instead modifies the output of the ESG parse.

- I heard that Edison invented the phonograph in 1877.
 - I heard that Edison invented a phonograph in 1877.
 - I heard Edison invented the phonograph in 1877.
 - I heard that Edison was inventing the phonograph in 1877.
 - I heard that the phonograph was invented by Edison in 1877.
- E.g, Have different meanings,, generate different ESG parse trees, yet reduce to the same PAS structure
 - Exact semantic meaning irrelevant since they all contain the same evidence to answer a question “Who invented the phonograph?”

PAS BUILDER

- Collapses the deep structure and surface structure into one
- Omits nodes
 - Auxiliary verbs
 - Nodes introducing Verb Phrases (e.g. “to”, “that”)
 - Determiners, except specially designated “high-semantics” determiners
 - Forms of “be” with no predicate
 - Forms of “be” for which the predicate is an adjective.
- Reduces part of speech taxonomy
- Goal: normalize results of ESG parse trees to better match candidate answer passages

PAS BUILDER

subj(n)	chandelier(1)	noun cn pl physobj artf
lconj	look(2,1,3)	verb vfin vpres pl sta
comp(a)	great(3,1,u)	adj erest
top	but(4)	verb vfin vpres pl cord
vadv	nowadays(5,6)	adv
rconj	do(6,1,9)	verb vfin vpres pl
vadv	not(7,6)	adv ppadv nounadv neg
vadv	usually(8,9)	adv
auxcomp(bin)	use(9,1,11,u)	verb vinf vpref ssa
ndet	these(10)	det pl def
obj(n)	item(11,u)	noun cn pl
comp(p)	from(12,17,13)	prep wh
objprep(n)	which(13,11,u)	noun pron wh
ndet	their(14)	det sg possdet
subj(n)	name(15,u,u)	noun cn sg langunit
nrel	be(16,15,17)	verb vfin vpres sg
pred(en)	derive(17,u,15,12)	verb ven vpass

```

chandelier (1)
look (2, subj:1, comp:3)
great (3)
but (4, lconj:2, rconj:9) [ top predicate]
nowadays (5)
not (7)
usually (8)
use (9, subj:1, obj:11, vadv:5, vadv:7,
    vadv:8)
item (11, nrel:17) [ determiner: these]
from (12, objprep:13)
which (13)
their (14)
name (15, ndet:14) [ determiner: their]
derive (17, obj:15, comp:12)
  
```

PATTERN-BASED RELATION EXTRACTION

- Identify semantic relations based on syntactic structure
 - E.g. authorOf, actorIn, bornOn
- Many different ways in which a relation can be expressed
- Therefore a need to abstract a generalized small set of rules to seek semantic relations
 - authorOf :: [Author] [WriteVerb] [Work]
 - authorOf :: [WriteVerb] -> subj -> [Author] & [WriteVerb] -> obj -> [Work]
 - authorOf :: [Author] -> Label -> ["of"-Arg] & ["of"-Arg] -> objprep -> [Work]

- (1) In 1936, he wrote his last play, "The Boy David"; an actress played the title role.
- (2) Born in Winsted, he practiced law in Connecticut before he wrote "Unsafe at Any Speed".
- (3) This "French Connection" actor coauthored the 1999 novel "Wake of the Perdido Star".
- (4) Walter Mosley penned this mystery about Detective Easy Rawlins searching for a woman in post-WWII L.A.
- (5) In December 1513, he wrote Francesco Vettori that he'd "composed a little work 'on pryncedoms'".
- (6) A "manly" 19th century realist, she penned works like "Adam Bede", "Felix Holt" and "Daniel Deronda".
- (7) Robert Louis Stevenson fell in love with Fanny Osbourne, a married woman, and later wrote this tale for her son.
- (8) This friend who refused to destroy Kafka's works wrote a historical novel on Tycho Brahe.
- (9) While living in Vermont, Kipling began writing this tale of an orphaned son of an Irish soldier in India.

USE IN ANSWERING QUESTION

- “Chandeliers look great but nowadays do not usually use these items from which their name is derived.”
 - Focus: “these items”, inferred from determiner “these”
 - Focus head: “items” -> AnswerType(item)
 - “use” obj = “these items” & “use” subj = “Chandeliers”
 - “use” has modifier “not”
 - Look for passages containing use with subject Chandelier with modifier not

EVALUTAION

- IBM-modified Deep Parsing is accurate
 - 92.0% accuracy on parsing of Jeopardy questions compared to 83.6% accuracy of popular Charniak Parser
 - 88.7% accuracy parsing Wikipedia compared to 81.1% accuracy of Charniak Parser
- Efficient
 - 5000 words per second on average laptop
 - ~50-100 times faster than Charniak Parser
 - Occupies ~5.7MB on disk
 - ~52MB memory footprint