Overview

1. Why combine deep and shallow processing?
2. Compatible semantics: initial examples
3. Composition
4. Experiments with RASP
5. Towards Robust MRS

Deep processing: advantages and problems

e.g., HPSG, LFG, TAG, CG etc

- Proper coverage of large proportion of phenomena is possible
- Detailed semantic representation
- Generation
- Not as slow as it used to be!

But:

- Not robust to unknown words, missing subcategorization information, ungrammatical input etc
- Not fast enough to process large volumes of text
- High level of ambiguity in longer sentences

Shallow processing

e.g., POS tagging, NP chunking, simple dependency structures.

- Fast
- Robust

But:

- Lack of lexical subcategorization information for open-class words precludes extraction of conventional semantic representations
- Long-distance dependencies
- Insufficient information to recover scope information
- Allows ungrammatical strings

Combining deep and shallow processing

- Shallow parse complete text, use deep parsing on selected sentences
- Shallow parser as preprocessor to guide deep parsing
- Question answering: deep parse questions, shallow parse answer texts
- Integrated shallow and deep parsing, identify some parts of sentences that should only be shallow parsed (Simone Teufel):

  We show that to substantially improve verb sense disambiguation it will be necessary to extract subcategorization information.

- Regeneration:

  Extraction of subcategorization information is a prerequisite for substantial improvement in verb sense disambiguation.

Interfacing via semantics

If shallow parsing could return underspecified semantic representations:

- Integrated parsing: shallow parsed phrases could be incorporated into deep parsed structures
- Deep parsing could be invoked incrementally in response to information needs
- Reuse of knowledge sources: information for further processing of shallow parsing might also be used on deep parser output (e.g., recognition of named entities, transfer rules in MT)
- Formal properties should be clearer, representations might be more generally usable

Simple example

Deep parser:

every fat cat sat on some mat

$$\text{every}(x, \text{fat}(x) \wedge \text{cat1}(x),$$
$$\quad \text{some}(y, \text{mat1}(y), \text{sit1}(e, x) \wedge \text{spast}(e) \wedge \text{on2}(e', e, y)))$$

$$\text{some}(y, \text{mat1}(y),$$
$$\quad \text{every}(x, \text{fat}(x) \wedge \text{cat1}(x), \text{sit1}(e, x) \wedge \text{spast}(e) \wedge \text{on2}(e', e, y)))$$

(The assumption of the 'extra' event in prepositions is motivated by predicative use after a copula.)

Question: how can we relate this to semantics from a shallow processor?
Extreme case: POS tagger

Semantic output from a POS tagger

POS tagger can be viewed as providing some semantic information:

- Lemmatization and morphology, partial disambiguation, BUT no relational information

every_DT0 fat_AJ0 cat_NN1 sat_VVD on_PRP some_DT0 mat_NN1

$$\text{every\_DT}(x1), \text{fat\_AJ}(x2), \text{cat\_N}(x3), \text{sit\_V}(e4), \text{past}(e4),$$
$$\text{on\_PRP}(e5), \text{some\_DT}(x6), \text{mat\_N}(x7)$$

Tag ‘lexicon’ (entries for tags, not lexemes):

DT0	lexrel_DT(x)	AJ0	lexrel_AJ(x)
NN1	lexrel_N(x)	PRP	lexrel_PRP(e)
VVD	lexrel_V(e), past(e)		

Can we manipulate syntax of semantics from deep processing to make the connection more obvious?

Modified syntax for deep representation

Flat semantics, underspecified scope (MRS):

$$h0 : \text{every}(x, h1, h2), h1 : \text{fat}(x), h1 : \text{cat1}(x),$$
$$h4 : \text{sit1}(e, x), h4 : \text{spast}(e), h4 : \text{on2}(e', e, y),$$
$$h5 : \text{some}(y, h6, h7), h6 : \text{mat1}(y)$$

Ignoring scope for now:

$$\text{every}(x), \text{fat}(x), \text{cat1}(x), \text{sit1}(e, x), \text{spast}(e), \text{on2}(e', e, y),$$
$$\text{some}(y), \text{mat1}(y)$$

Parsons' style representation:

$$\text{every}(x), \text{fat}(x), \text{cat1}(x), \text{sit1}(e), \text{spast}(e), \text{on2}(e'),$$
$$\text{some}(y), \text{mat1}(y),$$
$$ARG1(e, x), ARG1(e', e), ARG2(e', y)$$

Modified syntax for deep representation (continued)

Distinct variable names, plus equalities:

> every$(x1)$, fat$(x2)$, cat1$(x3)$, sit1$(e4)$, spast$(e4)$, on2$(e5)$,
> some$(x6)$, mat1$(x7)$,
> $ARG1(e4, x1)$, $ARG1(e5, e4)$, $ARG2(e5, x6)$,
> $x1 = x2, x2 = x3, x6 = x7$

Predicate naming convention:

every_DT$(x1)$, fat_AJ$(x2)$, cat_N$(x3)$, sit_V1$(e4)$, spast$(e4)$, on_PRP2$(e5)$,
some_DT$(x6)$, mat_N1$(x7)$,
$ARG1(e4, x1)$, $ARG1(e5, e4)$, $ARG2(e5, x6)$,
$x1 = x2, x2 = x3, x6 = x7$

Where sit_V1 $\sqsubset$ sit_V, spast $\sqsubset$ past, on_PRP2 $\sqsubset$ on_PRP,
mat_N1 $\sqsubset$ mat_N

Underspecification

Tagger output is an underspecified form of the deep representation.

Tagger output was:

every_DT($x1$), fat_AJ($x2$), cat_N($x3$), sit_V($e4$), past($e4$), on_PRP($e5$), some_DT($x6$), mat_N($x7$)

This can be converted into the deep representation by adding ARGn relations and equalities and specialising predicates.

Output from NP chunker is intermediate in specificity:

every_DT($x1$), fat_AJ($x2$), cat_N($x3$), sit_V($e4$), past($e4$), on_PRP($e5$), some_DT($x6$), mat_N($x7$), $x1 = x3, x2 = x3, x6 = x7$

RASP

- Robust accurate domain-independent statistical parsing (Briscoe and Carroll, 1995, 2002)
- Tokenizer, POS tagger, lemmatizer, tag-sequence grammar, statistical disambiguator
- Lexicon only contains POS tags:

  ```
  WORD VVD : V0[FIN +, AUX -, VFORM PAST, CONJ -].
  ```

  Rules mention SUBCAT, but not lexically instantiated:

  ```
  PSRULE S/np_vp : V2[WH -, INV -] --> N2 H1[FIN +].
  PSRULE V/np : V1 --> H0[VSUBCAT NP] N2.
  ```

- Doesn't do long-distance dependencies etc
- Intermediate depth: robust, but not super-fast.

Robust semantics from RASP

every fat cat sat on some mat

$\text{every\_DT}(x1), \text{fat\_AJ}(x2), \text{cat\_N}(x3), \text{sit\_V}(e4), \text{past}(e4), \text{on\_PRP}(e5),$
$\text{some\_DT}(x6), \text{mat\_N}(x7),$
$ARG1(e4, x1), ARG1(e5, e4), ARG2(e5, x6),$
$x1 = x2, x2 = x3, x6 = x7$

RASP output is almost equivalent to deep parsing under some circumstances

that milk every fat cat liked

$\text{that\_DT}(x1), \text{milk\_N}(x2), \text{every\_DT}(x3), \text{fat\_AJ}(x4), \text{cat\_N}(x5),$
$\text{like\_V}(e6), \text{past}(e6),$
$ARG1(e6, x5), x1 = x2, x3 = x4, x4 = x5$

Topicalization: can't retrieve relationship.

Specificity ordering

- Defined in terms of the syntax of the semantic representation, not the denotation
- Preliminary informal definition: a semantic representation A is an underspecified form of B if A can be converted to B by:
  1. adding argument-relations (e.g., ARG1)
  2. adding equalities between variables
  3. specialising predicates

Formal properties:

- Treat representations as made up of minimal units of semantic information
- Specificity as a partial order (cf (typed) feature structures)
- Defaults might be defined along the lines of YADU (Lascarides and Copestake 1999)

Parsons style semantics

Parsons (1991) — kick$(e) \wedge$ agent$(e, x) \wedge$ patient(e, y) rather than kick(e, x, y)
This means that *Kim kicked Sandy* implies *Kim kicked*

Dowty (1989) — proto-agent and proto-patient

Here: lack of lexicon in shallow semantics means we can't know the arity of open-class predicate, but Parsons style representation allows incremental addition of arguments. Use ARG1 etc rather than agent/patient. Underspecification of ARG-relations allows for dependency structures where we know there is a relation but can't be sure which one it is.

RMRS proper: argument/adjunct underspecification, use of labels/handles rather than event (cf Verbmobil VITs)

Underspecification of relations

e.g., sit_V1 $\sqsubset$ sit_V, spast $\sqsubset$ past

ARG1 $\sqsubset$ ARGN etc

$$\text{that\_DT}(x1), \text{milk\_N}(x2), \text{every\_DT}(x3), \text{fat\_AJ}(x4), \text{cat\_N}(x5),$$
$$\text{like\_V}(e6), \text{past}(e6),$$
$$ARG1(e6, x5), ARGN(e6, x2), x1 = x2, x3 = x4, x4 = x5$$

A refinement is to use a hierarchy of relations:
ARG2 $\sqsubset$ ARG2-4, ARG3 $\sqsubset$ ARG2-4,
ARG4 $\sqsubset$ ARG2-4, ARG2-4 $\sqsubset$ ARGN

Indexing schemes

Index variables according to string position (cf chart parsing/chart generation): variables have consistent names.

every fat cat sat on some mat

$$\text{every}(x1), \text{fat}(x2), \text{cat1}(x3), \text{sit1}(e4), \text{spast}(e4), \text{on2}(e5),$$
$$\text{some}(x6), \text{mat1}(x7)$$

1. Efficient comparison of semantic structures from the same string.
2. Relating semantic structures to parts of a string to allow deeper processing to be invoked.

An alternative is to index 'base' predications.

Lexical requirements

1. Consistent naming conventions
2. Deeper representations required to relate to shallower predicates if they make finer-grained distinctions:
   e.g., sit_V1 $\sqsubset$ sit_V
3. Consistent convention for ARG1 etc
4. May require lexicon for some closed class words for shallow/intermediate processing (e.g., don't want any semantics for some closed class words)

Composition

- Composition has to be formalised for shallow and intermediate parsing
- Consistent composition rules needed for tight integration of deep and shallow processing

LKB / RASP consistency:

- Deep structure composition expressed in TFSs for HPSG
- Semantic algebra used for MRS (Copestake, Lascarides and Flickinger, 2001) can be adjusted to robust semantics with RASP

Semantic composition in TFS for deep grammars

$$\begin{bmatrix} \text{SYN} & \begin{bmatrix} \textbf{np} \\ \text{HEAD } \textbf{noun} \\ \text{SUBJ } \langle\ \rangle \\ \text{COMPS } \langle\ \rangle \end{bmatrix} \\ \text{SEM} & \begin{bmatrix} \text{INDEX } \boxed{5}\,\textbf{ref-ind} \\ \text{LZT } \left\langle \begin{bmatrix} \text{RELN } \textbf{Kim} \\ \text{INSTANCE } \boxed{5} \end{bmatrix} \right\rangle \end{bmatrix} \end{bmatrix}$$

Kim

$$\begin{bmatrix} \text{SYN} & \begin{bmatrix} \text{HEAD } \textbf{verb} \\ \text{SUBJ } \left\langle \begin{bmatrix} \text{SYN } \textbf{np} \\ \text{SEM } \begin{bmatrix} \text{INDEX } \boxed{6} \end{bmatrix} \end{bmatrix} \right\rangle \\ \text{COMPS } \langle\ \rangle \end{bmatrix} \\ \text{SEM} & \begin{bmatrix} \text{INDEX } \boxed{15}\ \textbf{event} \\ \text{LZT } \left\langle \begin{bmatrix} \text{RELN } \textbf{SLEEP} \\ \text{ARG1 } \boxed{15} \\ \text{ARG2 } \boxed{6} \end{bmatrix} \right\rangle \end{bmatrix} \end{bmatrix}$$

sleeps

$$\begin{bmatrix} \text{SYN} & \begin{bmatrix} \text{HEAD } \textbf{verb} \end{bmatrix} \\ \text{SEM} & \begin{bmatrix} \text{INDEX } \boxed{2}\,\textbf{event} \\ \text{LZT } \boxed{10} \left\langle \begin{bmatrix} \text{RELN } \textbf{KIM} \\ \text{INSTANCE } \boxed{4} \end{bmatrix} \right\rangle + \boxed{11} \left\langle \begin{bmatrix} \text{RELN } \textbf{SLEEP} \\ \text{ARG1 } \boxed{2}\,\textbf{event} \\ \text{ARG2 } \boxed{4} \end{bmatrix} \right\rangle \end{bmatrix} \\ \text{HEAD-DTR.SEM} & \begin{bmatrix} \text{INDEX } \boxed{2} \\ \text{LZT } \boxed{11} \end{bmatrix} \\ \text{NON-HD-DTR.SEM.LZT } \boxed{10} & \end{bmatrix}$$

Kim sleeps

Example continued

- Sentence semantics equivalent to $Kim(x) \wedge sleep(e, x)$
- variable equivalence represented by coindexation
- coindexation in *Kim sleeps* by instantiating the SUBJ slot in *sleeps*
- **elementary predications** (EPs) represent individual predicate applications
  e.g., $\begin{bmatrix} \text{RELN} & \textbf{SLEEP} \\ \text{ARG1} & \boxed{15} \\ \text{ARG2} & \boxed{6} \end{bmatrix}$
- EPs accumulated by an append operation on the LZT feature (cf MRS): conjunction is implicit
- all signs have an INDEX (cf λ-variable, but not the same)

A semantic algebra

Kim : $[x_2]\{[]_{subj}, []_{comp}\}[Kim(x_2)]\{\}$
sleeps : $[e_1]\{[x_1]_{subj}, []_{comp}\}[sleep(e_1, x_1)]\{\}$
Kim sleeps : $[e_1]\{[]_{subj}, []_{comp}\}[sleep(e_1, x_1), Kim(x_2)]\{x_1 = x_2\}$

Semantic entities in algebra:

hook values that can fill a slot (i.e., variables are equated)

slots gaps corresponding to syntactic unsaturation (may be empty)

lzt bag of **elementary predications** (EPS)

equalities relate variables

$[e_1]\{[x_1]_{subj}, []_{comp}\}[sleep(e_1, x_1)]\{\}$

Composition

$$\begin{array}{ll} \text{Kim}: & [x_2]\{[]_{subj}, []_{comp}\}[\mathit{Kim}(x_2)]\{\} \\ \text{sleeps}: & [e_1]\{[x_1]_{subj}, []_{comp}\}[\mathit{sleep}(e_1, x_1)]\{\} \\ \hline \text{Kim sleeps}: & [e_1]\{[]_{subj}, []_{comp}\}[\mathit{sleep}(e_1, x_1), \mathit{Kim}(x_2)]\{x_1 = x_2\} \end{array}$$

Composition:

1. a (syntactically specified) slot in one structure (the **semantic head**) is filled by the hook of the other structure (by adding equalities)
2. lzts are appended
3. hook of phrase is the semantic head's hook
4. slots of phrase are unfilled slots of daughters

Access to semantics of sign during composition is only via hook and slots.

Semantic algebra and TFS 1

$$[e_1]\{[x_1]_{subj}, [\,]_{comp}\}[\mathit{sleep}(e_1, x_1)]\{\}$$

Feature structure:

$$\begin{bmatrix} \text{SYN} & \begin{bmatrix} \text{HEAD } \textbf{verb} \\ \text{SUBJ} \ \langle \begin{bmatrix} \text{SYN } \textbf{np} \\ \text{SEM } \begin{bmatrix} \text{INDEX } \boxed{6} \end{bmatrix} \end{bmatrix} \rangle \\ \text{COMPS} \ \langle\ \rangle \end{bmatrix} \\ \text{SEM} & \begin{bmatrix} \text{INDEX } \boxed{15} \ \textbf{event} \\ \text{LZT} \ \langle \begin{bmatrix} \text{RELN } \textbf{SLEEP} \\ \text{ARG1 } \boxed{15} \\ \text{ARG2 } \boxed{6} \end{bmatrix} \rangle \end{bmatrix} \end{bmatrix}$$

Semantic algebra and TFS 2

$$[x_2]\{[]_{subj}, []_{comp}\}[\mathit{Kim}(x_2)]\{\}$$

$$\begin{bmatrix} \text{SYN} & \begin{bmatrix} \textbf{np} \\ \text{HEAD } \textbf{noun} \\ \text{SUBJ} < > \\ \text{COMPS} < > \end{bmatrix} \\ \text{SEM} & \begin{bmatrix} \text{INDEX } \boxed{5}\,\textbf{ref-ind} \\ \text{LZT} < \begin{bmatrix} \text{RELN } \textbf{KIM} \\ \text{INSTANCE } \boxed{5} \end{bmatrix} > \end{bmatrix} \end{bmatrix}$$

$$[e_1]\{[]_{subj}, []_{comp}\}[\mathit{sleep}(e_1, x_1), \mathit{Kim}(x_2)]\{x_1 = x_2\}$$

$$\begin{bmatrix} \text{SYN} & \begin{bmatrix} \text{HEAD } \boxed{0}\ \textbf{verb} \end{bmatrix} \\ \text{SEM} & \begin{bmatrix} \text{INDEX } \boxed{2}\,\textbf{event} \\ \text{LZT } \boxed{10} < \begin{bmatrix} \text{RELN } \textbf{KIM} \\ \text{INSTANCE } \boxed{4} \end{bmatrix} > + \boxed{11} < \begin{bmatrix} \text{RELN } \textbf{SLEEP} \\ \text{ARG1 } \boxed{2}\,\textbf{event} \\ \text{ARG2 } \boxed{4} \end{bmatrix} > \end{bmatrix} \\ \text{HEAD-DTR.SEM} & \begin{bmatrix} \text{INDEX } \boxed{2} \\ \text{LZT } \boxed{10} \end{bmatrix} \\ \text{NON-HD-DTR.SEM.LZT } \boxed{11} & \end{bmatrix}$$

Simple semantic composition with RASP

- Slots are specified by rules rather than lexically
- Rules have semantic structure like phrases
- Algebra formalisation requires binary combination: rules are combined with daughters one-by-one
- Principles are otherwise much the same

$$
\begin{array}{ll}
\text{Kim}: & [x_1]\{\}[\mathit{Kim}(x_1)]\{\} \\
\text{sleeps}: & [e_2]\{\}[\mathit{sleep}(e_2)]\{\} \\
\text{S} \rightarrow \text{NP VP}: & []\{[x]_{NP}, [e]_{VP}\}[\mathit{ARG1}(e, x)]\{\} \\
\hline
\text{Kim sleeps}: & [e_2]\{\}[\mathit{sleep}(e_2), \mathit{Kim}(x_1), \mathit{ARG1}(e, x)]\{x_1 = x, e_2 = e\}
\end{array}
$$