Foundations of Language Science and Technology

Semantics 4a

Manfred Pinkal
Saarland University

Textual Inference and Logical Inference

P: *Several airlines reported cost increases*

H: *Several companies reported cost increases*

Example

P: *Several airlines polled saw costs grow more than expected.*
H: *Some companies reported cost increases.*

Atomic Edit	Lexical entailment	Sentence-level e.
SUB(*several, some*)	→ ⊏	→ ⊏
SUB(*airlines, companies*)	→ ⊏	→ ⊏
DEL(*polled*)	→ ⊏	→ ⊏
SUB(*saw, reported*)	→ ≡ ?	→ ≡
SUB(*costs, cost*)	→ ≡	→ ≡
SUB(*grow, increase*)	→ ≡	→ ≡
DEL(*more than expected*)	→ ⊏	→ ⊏

The effect of context: Monotonicity properties

P: *John bought a new convertible.*
H: *John bought a new car.*

P: *John didn't buy a new convertible.*
H: *John didn't buy a new car.*

P: *All airlines* reported cost increases.
P: *All companies* reported cost increases.

P: *All* airlines reported *extreme cost increases*.
P: *All* airlines reported *cost increases*.

What we need

- A method to find the best or most appropriate alignment/ sequence of edit steps between P and H.
- A general definition for entailment between expressions of arbitrary type.
- A method to identify the specific lexical entailment relations induced by specific SUB edits; DEL and INS induce ⊏ and ⊐, respectively.
- A method to determine monotonicity properties of contexts
- A compositional method to project entailment relations to the sentence level, taking monotonocity properties of context into account.
- A full specification of the join operation between entailment relations.

What we need

- A method to find the best or most appropriate alignment/ sequence of edit steps between P and H.
- A general definition for entailment between expressions of arbitrary type.
- A method to identify the specific lexical entailment relations induced by specific SUB edits; DEL and INS induce ⊏ and ⊐, respectively.
- A method to determine monotonicity properties of contexts
- A compositional method to project entailment relations to the sentence level, taking monotonocity properties of context into account.
- A full specification of the join operation between entailment relations.

General definition of entailment

- For sentences A, B $\in WE_t$:
  A ⊏ B iff A ⊨ B
- For proper nouns a, b $\in WE_e$:
  a ⊏ b iff a ⊐ b iff [[a]] = [[b]]
- For functional expressions α, β $\in WE_{<\sigma,\tau>}$:
  α ⊏ β iff for all $d \in D_\tau$: [[α]](*d*) ⊏ [[β]](*d*)

What we need

- A method to find the best or most appropriate alignment/ sequence of edit steps between P and H.
- A general definition for entailment between expressions of arbitrary type.
- A method to identify the specific lexical entailment relations induced by specific SUB edits.
- A method to determine monotonicity properties of contexts
- A compositional method to project entailment relations to the sentence level, taking monotonocity properties of context into account.
- A full specification of the join operation between entailment relations.

Lexical Entailment

- Assignment of lexical entailment uses features such as
  - WordNet synonymy (⊏ and ⊐), hyponymy (⊏), antonymy (neither ⊏ nor ⊐)
  - distributional similarity
  - part of speech (in particular: proper noun/ common noun/ pronoun)
  - string similarity (for pairs of proper nouns)
  - special logically fixed relations (*all* ⊏ *some*, *and* ⊏ *or*)
- Concrete assignment of entailment relations is done with a (decision tree) classifier.

What we need

- A method to find the best or most appropriate alignment/ sequence of edit steps between P and H.
- A general definition for entailment between expressions of arbitrary type.
- A method to identify the specific lexical entailment relations induced by specific SUB edits; DEL and INS induce ⊏ and ⊐, respectively.
- A method to determine monotonicity properties of contexts
- A compositional method to project entailment relations to the sentence level, taking monotonocity properties of context into account.
- A full specification of the join operation between entailment relations.

Monotonicity

- $\alpha \in WE_{<\sigma,\tau>}$ is upward monotonic, iff α denotes a function f such that

  for all $d, d' \in D_\sigma$: $f(d) \sqsubset f(d')$ iff $d \sqsubset$ **d'**.

- $\alpha \in WE_{<\sigma,\tau>}$ is downward monotonic, iff α denotes a function f such that

  for all $d, d' \in D_\sigma$: $f(d) \sqsubset f(d')$ iff $d \sqsupset$ **d'**.

Monotonicity, Examples

- Most verbs and common nouns are upward monotonic.

red:

upward monotonic, e.g.: *convertible* ⊏ *car*

red convertible ⊏ *red car*

big:

neither: *flea* ⊏ *animal*

big flea # big animal

doesn't :

downward monotonic: *walk* ⊏ *move*

doesn't walk ⊐ *doesn't move*

What we need

- A method to find the best or most appropriate alignment/ sequence of edit steps between P and H.
- A general definition for entailment between expressions of arbitrary type.
- A method to identify the specific lexical entailment relations induced by specific SUB edits; DEL and INS induce $\sqsubset$ and $\sqsupset$, respectively.
- A method to determine monotonicity properties of contexts
- A compositional method to project entailment relations to the sentence level, taking monotonocity properties of context into account.
- A full specification of the join operation between entailment relations.

Monotonicity, Examples

- Upward monotonic context: $\sqsubset \Rightarrow \sqsubset$, $\sqsupset \Rightarrow \sqsupset$, $\# \Rightarrow \#$
- Downward monotonic context: $\sqsubset \Rightarrow \sqsupset$, $\sqsupset \Rightarrow \sqsubset$, $\# \Rightarrow \#$
- Neither: $\sqsubset \Rightarrow \#$, $\sqsupset \Rightarrow \#$, $\# \Rightarrow \#$

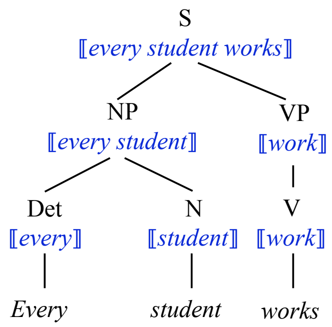
What we need

- A method to find the best or most appropriate alignment/ sequence of edit steps between P and H.
- A general definition for entailment between expressions of arbitrary type.
- A method to identify the specific lexical entailment relations induced by specific SUB edits; DEL and INS induce $\sqsubset$ and $\sqsupset$, respectively.
- A method to determine monotonicity properties of contexts
- A compositional method to project entailment relations to the sentence level, taking monotonocity properties of context into account.
- A full specification of the join operation between entailment relations.

The join operation

- $\sqsubset + \sqsubset = \sqsubset$
- $\sqsupset + \sqsupset = \sqsupset$
- All other combinations of $\sqsubset$, $\sqsupset$, and # yield #

Quantification



The Semantics of Quantified NPs

Every student works.

every-student': ((e,t),t) work': (e,t)

every-student'(work'): t

'*Every student*' denotes a second-order predicate that is true of a first-order predicate, if all students are in the denotation of that predicate.

More technically speaking, for $A \subseteq U_M$:

$[\![\text{every-student}']\!]^{M,g}(A)=1$ iff $V_M [\![\text{student}']\!] \subseteq A$

Simliarly for '*a student*' and '*no student*'

Examples

every student

$$\begin{bmatrix} \begin{bmatrix} j\to 1 \\ b\to 0 \\ m\to 1 \\ p\to 0 \\ s\to 1 \end{bmatrix} \to 1 \\ \begin{bmatrix} j\to 0 \\ b\to 1 \\ m\to 0 \\ p\to 0 \\ s\to 1 \end{bmatrix} \to 0 \\ \begin{bmatrix} j\to 1 \\ b\to 0 \\ m\to 0 \\ p\to 1 \\ s\to 1 \end{bmatrix} \to 0 \\ \ldots \end{bmatrix}$$

some students

$$\begin{bmatrix} \begin{bmatrix} j\to 1 \\ b\to 0 \\ m\to 1 \\ p\to 0 \\ s\to 1 \end{bmatrix} \to 1 \\ \begin{bmatrix} j\to 0 \\ b\to 1 \\ m\to 0 \\ p\to 0 \\ s\to 1 \end{bmatrix} \to 0 \\ \begin{bmatrix} j\to 1 \\ b\to 0 \\ m\to 0 \\ p\to 1 \\ s\to 1 \end{bmatrix} \to 1 \\ \ldots \end{bmatrix}$$

no student

$$\begin{bmatrix} \begin{bmatrix} j\to 1 \\ b\to 0 \\ m\to 1 \\ p\to 0 \\ s\to 1 \end{bmatrix} \to 0 \\ \begin{bmatrix} j\to 0 \\ b\to 1 \\ m\to 0 \\ p\to 0 \\ s\to 1 \end{bmatrix} \to 1 \\ \begin{bmatrix} j\to 1 \\ b\to 0 \\ m\to 0 \\ p\to 1 \\ s\to 1 \end{bmatrix} \to 0 \\ \ldots \end{bmatrix}$$

under the assumption that the denotation of *student* is {j, m}

Internal NP structure

Determiners denote functions from first-order predicates („student") to second-order predicates („every student") ; in other words: functions from first-order predicates to functions from first-order predicates to truth values.

every: ((e,t),((e,t),t)) student: (e,t)

every(student): ((e,t),t) work: (e,t)

every(student)(work): t

Semantically, *every* is a two-place second-order relation that takes two predicates as arguments and returns "true" if the denotation of the first is a subset of the denotation of the second predicate.

"every student works" is true iff the set of students is a subset of the set of working individuals.

Internal NP structure

- Other determiners, like „no“ or the indefinite article can be interpreted accordingly:

  V_M (every)(A)(B) = 1 iff $A \subseteq B$

  V_M (a)(A)(B) = 1 iff $A \cap B \neq \varnothing$

  V_M (no)(A)(B) = 1 iff $A \cap B = \varnothing$

- From these interpretations we can read off the monotonicity properties:
  - *a* is upward monotonic, *every* and *no* are downward monotonic (in their first argument).
  - *a student* and *every student* are upward monotonic, *no student* is downward monotonic.

Topics in Computational Semantics

- Refinement and compositional treatment of vector-space semantics
- Automatic acquisition of semantic resources (lexica, frame structures, scripts) from corpora
- Automatic acquisition of inference paraphrase and inference patterns from corpora
- Supervised, semi-supervised, unsupervised semantic processing
- Combining logic-based and distributional semantic methods