Language understanding and Bayesian inference

**Topic 3:** anchoring interpretation in world models

Daniel Lassiter
Stanford Linguistics
NASSLLI ‘14
UMD
course plan

1. probability and Bayesian inference
2. noisy channels, audience design, modularity & explaining-away
3. anchoring interpretation in world models
4. Bayesian pragmatics
plan

• build some simple modules & a fragment
  – a simple world model
  – a small lexicon interpreted in the world model
  – principles of syntactic, semantic composition
  – pragmatics of literal interpretation

• explore predictions about ambiguity resolution and plausibility
pragmatics as coordination

model-based inference of S beliefs, desires

\[ P_L(b_S, d_S|u) \propto P_S(u|b_S, d_S) \times P_L(b_S, d_S) \]

in principle, we would also have

\[ P_S(u|b_S, d_S) \propto P_L(b_S, d_S|u) \times P_S(u) \]

but this reasoning could go on forever …

(Lewis, 1969; Clark & Marshall, 1981)
interpretation for humans

choose a starting point, proceed by simulation

1. Literal interpretation uses a learned grammar and lexicon
2. Ss choose utterances by simulating literal listener
3. Pragmatic Ls infer $b_S d_S$ by simulating Ss

[... further iterations? ...]

(Goodman & Lassiter ’14, cf. Lewis ‘69, Franke ’09, ’12)
stochastic λ-calculus

• so far: specify generative models using math
• from here: mostly using pseudo-Church
• Church: Goodman et al. '08
  – pure functional lg. with probabilistic primitives
  – lazy evaluation
  – we’ll assume all functions implicitly memoized
• user-friendly version of stochastic λ-calculus
  http://probmods.org
world models

stochastically generate possible worlds
objects are placeholders, relations are functions

bob = gensym('bob')
mary = gensym('mary')
def flip (p = .5):
    random() < p
def love (x,y):
    flip(.5)

> bob
    #g31
> mary
    #g94
> love(mary, bob)
    False
> love(bob, mary)
    True
world models

let’s generate a possible world

```python
bob = gensym('bob')
mary = gensym('mary')
def flip (p = .5):
    random() < p
def love (x,y):
    flip(.5)

> bob
    #g401
> mary
    #g22
> love(mary, bob)
    True
> love(bob, mary)
    False
```
world models

and another

bob = gensym('bob')
mary = gensym('mary')
def flip (p = .5):
    random() < p
def love (x,y):
    flip(.5)

> bob
    #g19
> mary
    #g07
> love(mary, bob)
    True
> love(bob, mary)
    True
world models

distribution on evaluations implicitly represents joint distribution

```python
bob = gensym('bob')
mary = gensym('mary')
def flip(p = .5):
    random() < p
def love(x,y):
    flip(.5)
> bob
    #g401
> mary
    #g22
> love(mary, bob)
    False
> love(bob, mary)
    False
```
world models

auto-marginalization: ask about one variable, others are computed as needed but not returned

```
bob = gensym('bob')
mary = gensym('mary')
def flip (p = .5):
    random() < p
def love (x,y):
    flip(.5)

> bob  
    #g401
> mary  
    #g22
> love(mary, bob)  
    False
> love(bob, mary)  
    False
```
tug-of-war

simple but rich domain: players, teams, strength, total strength => pulling => winners, ...

```
players = [Bob, Mary, Bill, ...]
teams = [Team1, Team2, ...]
def gender (player):
    if flip() ‘male’ else ‘female’
def genderStrength (gender):
    Gaussian(0,2)
def strength (player):
    genderStrength(gender(player)) + Gaussian(0,1)
```
tug-of-war

matches = [match1, match2, match3, …]
def teamSize(team):
    uniform([2,4,6])
def playersOnTeam (team):
    draw(teamSize(team), players)
def pulling(team):
    sum(map(strength, playersOnTeam(team)))
def winner(match):
    pulling(teamsInMatch(match)[0]) > pulling(teamsInMatch(match)[1])
def query (model, QUD, condition):
    ... sample world w from model ...
    if ... condition is true in w ...
        return ... answer to QUD in w ...
    else:
        return query(model, QUD, condition)  # recurse

use to model
•  conditional reasoning
•  Bayesian update
example

observation: winner([Bob], [Jim]) == [Bob]
• How strong is Bob?

```query(ToW-model,          # model
       strength(Bob),      # QUD
       winner([Bob], [Jim]) == [Bob])  # condition
```

1. generate a ToW-world \( w \)
2. check that team [Bob] beat team [Jim] in \( w \)
3. if so, return Bob’s strength in \( w \)
4. otherwise, repeat until condition is satisfied
update: Jim is the weakest player in the tournament
• How strong is Jim?

query(ToW-model,
    strength(Bob),
    winner([[Bob], [Jim]]) == [Bob] and
    all(map(\lambda x. strength(x) > strength(Jim), other-players))))

Inference is non-monotonic
Inferred strength

Bob's strength

No observations (prior)
Observation: Bob beats Jim
Observation: Bob beats Jim, Jim weak
model-theoretic semantics

- words, sentences denote objects in a model
  - names pick out objects (elements of $D_e$)
  - properties: functions from $D_e$ to $\{0,1\}$
  - relations: functions from individuals to properties
  - etc.

- we’ll interpret into a stochastic world model.
  - cognitive interpretation of MTS
  - intensionality everywhere but implicit
role of SLC

• Montague (1973): $\lambda$-calculus as intermediate language is for convenience
• For us, stochastic $\lambda$-calculus is essential: common language for meanings, world models
• word meaning is ‘grounded’ in rich intuitive theories of the world
def literal-listener(utt, world-model, QUD):
    return query(world-model, QUD, meaning(utt) == True)
basic categorial grammar

• Basic categories **BCAT**: \{S, NP, N, CP, \ldots\}

• Complex categories **CAT**:
  – all basic categories are in CAT
  – If A and B are in CAT then
    • A/B is in CAT
    • A\B is in CAT
  – Nothing else in in CAT.

• **Combination:** \[ [A/B \; B]_A \quad [B \; A\B]_A \]
a simple lexicon

<table>
<thead>
<tr>
<th>Word</th>
<th>Syn</th>
<th>Sem</th>
</tr>
</thead>
<tbody>
<tr>
<td>'Bob'</td>
<td>NP</td>
<td>Bob</td>
</tr>
<tr>
<td>'Mary'</td>
<td>NP</td>
<td>Mary</td>
</tr>
<tr>
<td>'Team1'</td>
<td>NP</td>
<td>Team1</td>
</tr>
<tr>
<td>'man'</td>
<td>N</td>
<td>(\lambda x . \text{gender}(x) == \text{‘male’})</td>
</tr>
<tr>
<td>'loves'</td>
<td>((S\backslash NP)/NP)</td>
<td>(\lambda x \lambda y . \text{love}(y,x))</td>
</tr>
<tr>
<td>'won'</td>
<td>((S\backslash NP)/NP)</td>
<td>(\lambda x \lambda y . \text{winner}(x) == y)</td>
</tr>
<tr>
<td>'every'</td>
<td>((S/(S\backslash NP))/N)</td>
<td>(\lambda P \lambda Q . \text{all}(\text{map}(\lambda \ldots, D_e)))</td>
</tr>
</tbody>
</table>

...
recursive stochastic composition

def meaning(utt):
    p = randomParse(randomSegmentation(utt))
    if (atomic(p)):
        return randomTypeShift(lexicon(p))  # TBD
    else:
        left = randomTypeShift(p[0])
        right = randomTypeShift(p[1])
        if rightCombines(left, right):  # if gr. permits rCombination
            return combine(p[0], p[1])
        else if leftCombines(left, right) ......  # etc.
        else:
            return False  # falsity or ungrammaticality
def randomTypeShift(expression):
    if flip():
        return expression
    else:
        shifter = uniform-draw(LIFT, GEACH, AR1, AR2, ...)
        if flip():
            return shifter(expression)  # syntactic, semantic effects
        else:
            return randomTypeShift(shifter(expression))

stochastic recursion => pref. for interpretation in low types

(Parthee & Rooth, 1983)
def literal-listener(utt, world-model, QUD):
    return query(world-model, QUD, meaning(utt) == True)

test utterances for truth against simulated world predictions:

• interpretation is non-monotonic, like inference
• ambiguities resolved according to plausibility
• ...
non-monotonic interpretation

1. “Bob was on Team 1.”
2. “Jim was on Team 2.”
3. “Team 1 played Team 2.”

query (ToW-model,
  strength(Bob),
  meaning(1) == True and
  meaning(2) == True and
  meaning(3) == True )
non-monotonic interpretation

1. “Bob was on Team 1.”
2. “Jim was on Team 2.”
3. “Team 1 played Team 2.”
4. “Team 1 beat Team 2.”

query (ToW-model,
  strength(Bob),
  meaning(1) == True and
  meaning(2) == True and
  ... )
non-monotonic interpretation

1. “Bob was on Team 1.”
2. “Jim was on Team 2.”
3. “Team 1 played Team 2.”
4. “Team 1 beat Team 2.”
5. “Jim is the weakest player.”

query (ToW-model, 
  strength(Bob), 
  meaning(1) == True and 
  meaning(2) == True and 
  meaning(3) == True)

Bob's strength
def literal-listener(utt, world-model, QUD):
    return query(world-model, QUD, meaning(utt) == True)

• generate parses and interpretations stochastically
• condition on truth in simulated world
• reject if false or syntactically ill-formed
• plausible interpretations accepted more often
  – and the parses that generated them
ambiguity

expected typology of ambiguities
  – phonological
  – syntactic
  – compositional semantics
  – lexical
  – conceptual
lexicon(‘foo’)

- lexical ambiguity: stochastically return 1 of 2 deterministic functions
- conceptual ambiguity: deterministically return a stochastic function
- mixed: stochastically return a stochastic function

impossible to distinguish on linguistic grounds?
lexical ambiguity

1. The otter went to the bank.
2. The businessman went to the bank.
3. Mary went to the bank.

– out of context, Ls prefer most frequent meaning
– contextual plausibility modulates this preference

(Simpson, 1994)

highly compatible with Bayesian interpretation
syntactic ambiguity

The teachers taught by the Berlitz method ...

1. and loved it  
2. learned a lot

preference for main verb interpretation (1)

(Crain & Steedman 1985, etc.)
syntactic ambiguity

The students taught by the Berlitz method ...

1. and loved it
2. learned a lot

preference for RC interpretation (2)

- most work in this area on algorithmic questions
- we make no direct predictions about processing
- focus on understanding why plausibility matters
quantifier scope ambiguity

• generate however you like
  – if using CG, with type-shifting: Hendriks, Barker
• how do we **choose** an interpretation?
• preference should track probability of truth

1. A doctor treated every patient
   • how many doctors? how many patients?

2. A guard stood in front of every building
quantifier scope ambiguity

“most players played in some match”

• **SWS**: for most x, x played in some match
• **OWS**: for some match y, most players played in y
quantifier scope ambiguity

“most players played in some match”

• Fix 12 players in total, “most” > 6
• Assume team sizes 1-6 equally likely a priori
• What do you infer about team size with
  – 1 match?
  – 100 matches?
quantifier scope ambiguity

“most players played in some match”
next: recursive pragmatics

- speaker model that calls literal-listener()
- pragmatic listener that calls speaker()
  - reason about S’s motivations for producing $u$
  - condition on event of utterance, not its truth
- case studies:
  - quantity implicature
  - context-sensitivity and gradable adjectives