CS388: Natural Language Processing
Lecture 14: Semantics II / Seq2seq I

Greg Durrett

Administrivia

› Graham Neubig (CMU) talk this Friday at 11am in 6.302.
   “Towards Open-domain Generation of Programs from Natural Language”

› Mini 2 graded by the end of the week

› Project 2 out by Thursday

Recall: Parsers to Logical Forms

Recall: CCG

› Steedman+Szabolcsi 1980s: formalism bridging syntax and semantics

› Syntactic categories (for this lecture): S, NP, “slash” categories
  S\NP: “if I combine with an NP on my left side, I form a sentence” — verb
  (S\NP)/NP: “I need an NP on my right and then on my left” — verb
  with a direct object

General rules:

sings(e470) & dances(e470)

S

NP

VP

λy. sings(y) & dances(y)

S\NP

VP

CC

and

VP

VBPsinger

VBPdances

λy. sings(y)

λy. dances(y)

Eminem

sings

Oklahoma

borders

Texas
This Lecture

- Lambda-DCS: more lightweight than CCG
- Seq2seq models
- Seq2seq models for semantic parsing

Lambda-DCS

- Dependency-based compositional semantics — original version was less powerful than lambda calculus, lambda-DCS is as powerful
- Designed in the context of building a QA system from Freebase
- Freebase: set of entities and relations
  - DateOfBirth  
  - PlaceOfBirth  
  - CapitalOf  
  - PlaceOfBirth.Seattle
- Looks like a tree fragment over Freebase, denotes the set of people born in Seattle, no explicit variables

\[
\lambda x. \lambda y. \text{PlaceOfBirth}(x,y) \\
\lambda x. \text{PlaceOfBirth}(x,\text{Seattle})
\]

Liang et al. (2011), Liang (2013)
Lambda-DCS

March 15, 1961
Bob Cooper
Washington
Liang et al. (2011), Liang (2013)
Alice Smith
PlaceOfBirth
Seattle
Liang et al. (2011), Liang (2013)

“list of scientists born in Seattle”
Scientist
Profession
Profession.Scientist \land
PlaceOfBirth
PlaceOfBirth.Seattle

Execute this fragment against Freebase, returns Alice Smith (and others)

Parsing into Lambda-DCS

Derivation \( d \) on sentence \( x \):
- No more explicit syntax in these derivations like we had in CCG
- Everything is a set, sets combine in a few ways
- Building the lexicon: more sophisticated process than GENLEX, but can handle thousands of predicates

Log-linear model with features on rules:

\[
P(d|x) \propto \exp \left( \sum_{r \in d} f(r, x) \right)
\]
Berant et al. (2013)

Learn from derivations: standard supervised learning, maximize probability of correct derivation

\[
\mathcal{L}(\theta) = \sum_{i=1}^{n} \log P(d_i^* | x_i)
\]

Problem: supervision looks like “Where was Barack Obama born” — “Hawaii” without a derivation

Learn just from question-answer pairs: maximize the likelihood of the right denotation \( y^* \) with the derivation \( d \) marginalized out

\[
\mathcal{L}(\theta) = \sum_{i=1}^{n} \log \sum_{d: [d]|_K = y_i^*} P(d | x_i)
\]

Approx procedure: for each example:
- Run beam search to get a set of derivations
- Let \( d \) = highest-scoring derivation in the beam
- Let \( d^* \) = highest-scoring derivation in the beam with correct denotation
- Do a structured perceptron update towards \( d^* \) away from \( d \)
Berant et al. (2013)
**Learning**

- Each vertical slice is the beam for one example. Green = correct denotation

- Only a small number of questions are even reachable by beam search initially (but some questions are very easy so even a totally untrained model can answer them)
- During training, more and more “good” derivations surface and will result in model updates

Berant et al. (2013)

**Motivation**

- Parsers have been pretty hard to build...
  - Constituency/graph-based: complex dynamic programs
  - Transition-based: complex transition systems
  - CCG/semantic parsers: complex syntax/semantics interface, challenging inference, challenging learning
- For semantic parsing in particular: bridging the syntax-semantics divide results in structural weirdnesses in parsers
- Encoder-decoder models can be a lot more uniform — we’ll come back to this later in the lecture

**Encoder-Decoder Models**

- Only a small number of questions are even reachable by beam search initially (but some questions are very easy so even a totally untrained model can answer them)
- During training, more and more “good” derivations surface and will result in model updates

Berant et al. (2013)

**Motivation**

- Parsers have been pretty hard to build...
  - Constituency/graph-based: complex dynamic programs
  - Transition-based: complex transition systems
  - CCG/semantic parsers: complex syntax/semantics interface, challenging inference, challenging learning
- For semantic parsing in particular: bridging the syntax-semantics divide results in structural weirdnesses in parsers
- Encoder-decoder models can be a lot more uniform — we’ll come back to this later in the lecture

**Encoder-Decoder**

- Encode a sequence into a fixed-sized vector

  - le  film  était  bon [STOP]
  - the movie was great

- Now use that vector to produce a series of tokens as output from a separate LSTM decoder

Sutskever et al. (2014)
Encoder-Decoder

- Generate next word conditioned on previous word as well as hidden state

- W size is $|\text{vocab}| \times |\text{hidden state}|$, softmax over entire vocabulary

- During inference: need to compute the argmax over the word predictions and then feed that to the next RNN state

- Need to actually evaluate computation graph up to this point to form input for the next state

- Decoder is advanced one state at a time until [STOP] is reached

Model

- Generate next word conditioned on previous word as well as hidden state

- $P(y_n|x, y_1, \ldots, y_{n-1}) = \text{softmax}(W\bar{h})$

- $P(y|x) = \prod_{i=1}^{n} P(y_i|x, y_1, \ldots, y_{i-1})$

Inference

- Generate next word conditioned on previous word as well as hidden state

- During inference: need to compute the argmax over the word predictions and then feed that to the next RNN state

- Need to actually evaluate computation graph up to this point to form input for the next state

- Decoder is advanced one state at a time until [STOP] is reached

Implementing seq2seq Models

- Encoder: consumes sequence of tokens, produces a vector. Analogous to encoders for classification/tagging tasks

- Decoder: separate module, single cell. Takes two inputs: hidden state (vector $h$ or tuple $(h, c)$) and previous token. Outputs token + new state
Training

Training: Scheduled Sampling

Implementations Details

Implementations Details (cont’d)
**Beam Search**

- Maintain decoder state, token history in beam
- Do not max over the two film states! Hidden state vectors are different

**Seq2seq Semantic Parsing**

**Semantic Parsing as Translation**

“what states border Texas”

\[
\lambda x ( \text{state}(x) \text{ and border}(x, \text{e89}))
\]

- Write down a linearized form of the semantic parse, train seq2seq models to directly translate into this representation
- What are some benefits of this approach compared to grammar-based?
- What might be some concerns about this approach? How do we mitigate them?

Jia and Liang (2015)

**Handling Invariances**

“what states border Texas”

“what states border Ohio”

- Parsing-based approaches handle these the same way
- Possible divergences: features, different weights in the lexicon
- Can we get seq2seq semantic parsers to handle these the same way?
- Key idea: don’t change the model, change the data
- “Data augmentation”: encode invariances by automatically generating new training examples
Data Augmentation

Jia and Liang (2015)

Examples

"what states border texas ?

answer (NV, (state (V0), next_to (V0, NV), const (V0, stateid (texas))))

\(\text{"what is the highest mountain in ohio ?"} \)

answer (NV, highest (V0, mountain (V0), loc (V0, NV), const (V0, stateid (ohio))))

Rules created by ABSENTITIES

ROOT -> "what states border STATEID ?"

answer (NV, (state (V0), next_to (V0, NV), const (V0, stateid (STATEID))))

STATEID -> "texas", "texas"

ROOT -> "what is the highest mountain in STATEID ?"

answer (NV, highest (V0, mountain (V0), loc (V0, NV),

const (V0, stateid (STATEID))))

STATEID -> "ohio", "ohio"

- Lets us synthesize a “what states border ohio ?” example
- Abstract out entities: now we can “remix” examples and encode invariance to entity ID. More complicated remixes too

Semantic Parsing as Translation

Jia and Liang (2015)

Three forms of data augmentation all help

Results on these tasks are still not as strong as hand-tuned systems from 10 years ago, but the same simple model can do well at all problems

Semantics Parsing as Translation

Prolog

\(\text{Lambda calculus}\)

\(\text{Other DSLs}\)

- Handle all of these with uniform machinery!

Jia and Liang (2015)

<table>
<thead>
<tr>
<th>Geo</th>
<th>ATIS</th>
</tr>
</thead>
<tbody>
<tr>
<td>84.6</td>
<td>84.2</td>
</tr>
</tbody>
</table>

Table: Geo vs. ATIS

Previous Work

- Zettlemoyer and Collins (2007)
- Kwiatkowski et al. (2010)
- Liang et al. (2011)
- Kwiatkowski et al. (2011)
- Poorn (2013)
- Zhao and Huang (2015)

Our Model

- No Recombination
- ABSENTITIES
- ABSWHOLEPHRASES
- CONCAT-2
- CONCAT-3
- AWP + AE
- AE + C2
- AWP + AE + C2
- AE + C3

Locascio et al. (2016)

- Can use for other semantic parsing-like tasks
- Predict regex from text

Problem: requires a lot of data: 10,000 examples needed to get ~60% accuracy on pretty simple regexes
SQL Generation

- Convert natural language description into a SQL query against some DB
- How to ensure that well-formed SQL is generated?
  - Three seq2seq models
- How to capture column names + constants?
  - Pointer mechanisms

Zhong et al. (2017)

Attention

“what states border Texas” → lambda x ( state (x) and border (x, e89) )

- Orange pieces are probably reused across many problems
- Not too hard to learn to generate: start with lambda, always follow with x, follow that with paren, etc. This is a common question
- LSTM has to remember the value of Texas for 13 steps!
- Next lecture: attention mechanisms that let us “look back” at the input to avoid having to remember everything

Takeaways

- Lambda-DCS is a more lightweight formalism than lambda calculus
- Rather than combining syntax and semantics like in CCG, we can either parse to semantic representations directly or generate them with seq2seq models
- Seq2seq models are a very flexible framework, some weaknesses can potentially be patched with more data