AMR dependency parsing with a typed semantic algebra

Jonas Groschwitz*, Matthias Lindemann*, Meaghan Fowlie*, Mark Johnson^, Alexander Koller*

*Saarland University          ^Macquarie University

ACL 2018
Melbourne, Australia
July 17
The witch tried to cast a spell
The witch tried to cast a spell
Classic AMR parser (e.g. JAMR 2014)

Step 1: Predict nodes

try

witch

spell

Step 2: Predict edges

try

witch

ARG0

ARG1

spell

ARG1

ARG0

cast

The witch tried to cast a spell
Not just nodes and edges
Not just nodes and edges

Control verb

Noun

Transitive verb
Hidden compositional structure

Principle of compositionality: the meaning of a complex expression is determined by the meanings of its constituent expressions and the rules used to combine them.
Hidden compositional structure

Principle of compositionality: the meaning of a complex expression is determined by the meanings of its constituent expressions and the rules used to combine them.
Hidden compositional structure

Principle of compositionality: the meaning of a complex expression is determined by the meanings of its constituent expressions and the rules used to combine them.

- Widely accepted in linguistics, long history (Frege 1800s)
- Use this knowledge to guide machine learning!
The witch tried to cast a spell
The witch tried to cast a spell
The witch tried to cast a spell.

easy (easier)

equivalent

dependencies!

The witch tried to cast a spell
The witch tried to cast a spell.
Apply-Modify (AM) Algebra

G. et al, IWCS 2017

*HR algebra, Courcelle & Engelfriet 2012
Apply-Modify (AM) Algebra

G. et al, IWCS 2017

- Empty argument slots are labeled with sources* S,O,… (subject, object,…)

*HR algebra, Courcelle & Engelfriet 2012
Apply-Modify (AM) Algebra

G. et al, IWCS 2017

- Empty argument slots are labeled with sources* S,O,… (subject, object,…)
- Have ‘apply’ operation for each source, e.g. APP₀

*HR algebra, Courcelle & Engelfriet 2012
Typed AM Algebra

S

ARG0

Has type [S]

cast

ARG1

spell

witch

cast

try

spell
Typed AM Algebra

Object must have type [S]

Has type [S]
Typed AM Algebra

Object must have type [S]

Has type [S]

Matching sources automatically merge
Apply-Modify Algebra

The witch tried to cast a spell
Types control reentrancies

Has type [S]

*cast to sleep
Types control reentrancies

Object must have type [ ]

Has type [S]

*cast to sleep
Types control reentrancies

Appo

Object must have type [S]

Has type [[]]

*tried to witch
AM Dependency Trees

dependencies define operations, but not their order

The witch cast a spell

witch

cast

spell

Appo

Apps

ARG0

ARG1
AM Dependency Trees

dependencies define operations, but not their order

The witch cast a spell

here: order does not matter
The witch tried to cast a spell

here: need APP₀ before APPₛ to get reentrancies
AM Dependency Trees

The witch tried to cast a spell

here: need APP₀ before APPₛ to get reentrancies

• Always need to resolve reentrancies first
• Types encode reentrancies
  ➡ use type system to determine operation order
AM Dependency Trees

The witch tried to cast a spell

Building instructions for an AMR that we know how to predict
The witch tried to cast a spell.
Model

1. Supertagging: score graph fragments for each word

   ![Diagram of supertagging]

2. Dependency model: score operations

   ![Diagram of dependency model]

3. Decoding: find highest-scoring well-typed tree

   ![Diagram of decoding]
1. Supertagging

E.g. Lewis et al. (2014) for CCG

The witch tried spell

probability distribution over graph lexicon

bidirectional LSTM

word embeddings

train to predict
2. Dependency Model

Kiperwasser & Goldberg (2016) for syntactic dependencies

The witch tried spell

word embeddings

bidirectional LSTM

probability distribution over operations

train to predict

Apps

witch tried

P_{2\rightarrow 1}
The witch cast a spell
The witch cast a spell

**AMR Corpus**

**Required training data**

**Heuristics**

- Alignments
- Attaching edges
- Source names
- Source annotations
AMR Corpus

The witch cast a spell

Heuristics

- Alignments
- Attaching edges
- Source names
- Source annotations

Required training data

The witch cast a spell
3. Decoding

Find the **best well-typed** dependency tree
3. Decoding

Find the **best well-typed** dependency tree

- ill-typed trees do not evaluate to AMRs
- ill-typed trees do not match our linguistic intuitions
3. Decoding

Find the **best well-typed** dependency tree

- ill-typed trees do not evaluate to AMRs
- ill-typed trees do not match our linguistic intuitions
- Exact typed decoding is NP-hard
- Untyped decoding: 74% of trees are ill-typed
3. Decoding

Find the **best well-typed** dependency tree

- ill-typed trees do not evaluate to AMRs
- ill-typed trees do not match our linguistic intuitions
- Exact typed decoding is NP-hard
- Untyped decoding: 74% of trees are ill-typed

➤ Approximate decoders
Approximate decoders

**A:** Fixed tree

A diagram is shown with labeled nodes for "The," "witch," "cast," "a," and "spell."
Approximate decoders

A: Fixed tree

1. Fix unlabeled tree

2. Label tree, with type checking
Approximate decoders

A: Fixed tree

1. Fix unlabeled tree

2. Label tree, with type checking

B: Projective: can only combine adjacent constituents

"CKY parsing with types as nonterminals"
Results
Classic AMR parser (*graph decoder*)

**Step 1: Predict nodes**
- try
- spell
- witch
- cast

**Step 2: Predict edges**
- try
- spell
- witch
- cast

ARG0

ARG1
## Results

<table>
<thead>
<tr>
<th>Model</th>
<th>Method</th>
<th>Smatch score</th>
</tr>
</thead>
<tbody>
<tr>
<td>JAMR (Flanigan et al. 2016)</td>
<td>graph decoder</td>
<td>67</td>
</tr>
<tr>
<td>Foland &amp; Martin 2017</td>
<td>graph decoder</td>
<td>70.7</td>
</tr>
<tr>
<td>van Noord &amp; Bos 2017</td>
<td>neural seq2seq</td>
<td>68.5</td>
</tr>
<tr>
<td>Lyu &amp; Titov (ACL 2018)</td>
<td>graph decoder</td>
<td><strong>73.7</strong></td>
</tr>
<tr>
<td><strong>Our baseline</strong></td>
<td>graph decoder</td>
<td>66.1</td>
</tr>
<tr>
<td><strong>Our projective decoder</strong></td>
<td></td>
<td><strong>70.2</strong></td>
</tr>
<tr>
<td><strong>Our fixed tree decoder</strong></td>
<td></td>
<td><strong>70.2</strong></td>
</tr>
</tbody>
</table>

Dataset: LDC2015E86
Conclusion

• We built a competitive compositional AMR parser

• Clear avenues to improvement
  • Update to recent advancements in training regimen (e.g. Lyu & Tivov 2018)
  • Look into specific phenomena, e.g.
    • anaphora
    • ellipsis
  • Future work: extend method to other formalisms
Conclusion

- We built a competitive compositional AMR parser
- Clear avenues to improvement
  - Update to recent advancements in training regimen (e.g. Lyu & Tivov 2018)
  - Look into specific phenomena, e.g.
    - anaphora
    - ellipsis
- Future work: extend method to other formalisms