Joint A* CCG Parsing and Semantic Role Labelling

Mike Lewis, Luheng He, Luke Zettlemoyer
University of Washington
Semantic Role Labelling

John denied the report

John refused to deny the report

John refused to confirm or deny the report
Semantic Role Labelling

John denied the report

John refused to deny the report

John refused to confirm or deny the report
Semantic Role Labelling

1. John **denied** the report
2. John **refused to** **deny** the report
3. John **refused to confirm or** **deny** the report
Semantic Role Labelling

John denied the report

John refused to deny the report

John refused to confirm or deny the report
Semantic Role Labelling

John wanted to deny the report

John wanted her to deny the report
Semantic Role Labelling
Joint Modeling

- **Improve semantic role labelling**
  - Avoid pipeline errors

- **Improve syntactic parsing**
  - Reduce sparsity by abstracting over syntactic realizations
Joint vs. Pipelines

F1
Joint Modeling

- Semantic dependencies can span arbitrarily many syntactic dependencies
  - Difficult to parse jointly with dynamic programs
  - Syntax features are sparse

The reports that he refused to confirm or deny
This talk

1. Joint model for CCG parsing and SRL
2. Efficient A* parsing algorithm
This talk

1. Joint model for CCG parsing and SRL

2. Efficient A* parsing algorithm
CCG Parsing

John NP confirmed (S\NP)/NP the report NP

S\NP

S
CCG Parsing

John confirmed (S\NP)/NP

the report NP

(S\NP)/NP

λyλx . confirm → x ∧ confirm → y

(S\NP)/NP

λx . confirm → report ∧ confirm → x

S\NP

(S\NP)/NP

λyλx . confirm → x ∧ confirm → report

S

confirm → john ∧ confirm → report
CCG Parsing

John confirmed the report

\[ (S\text{NP})/NP \rightarrow (S\text{NP})/NP \]

\[ \lambda y \lambda x . \text{confirm} \rightarrow x \land \text{confirm} \rightarrow y \]
John
NP
wanted
(S\NP)/(S\NP)
\lambda p \lambda x . p(x)
to confirm
(S\NP)/NP
\lambda y \lambda x . \text{confirm} \rightarrow x \land \text{confirm} \rightarrow y
the report
NP
\lambda x . \text{confirm} \rightarrow \text{report} \land \text{confirm} \rightarrow x
CCG Parsing

wanted (S\NP)/(S\NP) \lambda p \lambda x . p(x)

to confirm (S\NP)/(S\NP) \lambda y \lambda x . S \lambda x .

the report (S\NP)/(S\NP) \lambda x . S \lambda x .

confirm S
CCG Parsing

John (S\NP)\(\)/(S\NP)

\[\lambda y . \text{confirm} \rightarrow y \quad \lambda x . \text{confirm} \rightarrow x \quad \lambda p . p(x)\]

the report (S\NP)/NP

\[\lambda x . \text{confirm} \rightarrow \text{report} \land \text{confirm} \rightarrow x\]

\[\lambda x . \text{confirm} \rightarrow \text{report} \land \text{confirm} \rightarrow x\]

S

\[\text{confirm} \rightarrow \text{john} \land \text{confirm} \rightarrow \text{report}\]
John wanted her to confirm the report.

\[
\lambda \lambda \lambda x . p(y) \\
\Lambda \Lambda \Lambda y . \text{confirm} \rightarrow x \land \text{confirm} \rightarrow y \\
\Lambda x . \text{confirm} \rightarrow \text{report} \land \text{confirm} \rightarrow x
\]
John wanted her to confirm the report.

\[
\begin{align*}
\text{John} & \quad \text{NP} \\
\text{wanted} & \quad \frac{(\text{S}\slash \text{NP})}{(\text{S}\slash \text{NP})} \quad \text{NP} \\
\text{her} & \quad \text{NP} \\
\text{to confirm} & \quad (\text{S}\slash \lambda x .) \\
\text{the report} & \quad \text{NP}
\end{align*}
\]
John wanted her to confirm the report.
Joint CCG/SRL Parsing

John refused to confirm or deny the report

- All information for creating and propagating dependencies is in the lexicon
- Joint CCG/SRL parsing is just labelling CCG dependencies
Semantic Role Labelling

He opened the door

ARG0

ARG1

The door opened

ARG1
Joint Model

Global log-linear model over joint space of CCG parses $y$ with semantic roles $l(y)$

$$p(y, l(y)|x) \propto e^{\theta \cdot \phi(x, y, l(y))}$$
Training

Learn latent CCG that recovers SRL

He opened the door

ARG0

ARG1
Training

Learn latent CCG that recovers SRL

- Generate *consistent* CCG/SRL parses for training sentences
Training

Learn latent CCG that recovers SRL

• Mark subset as correct, based on semantic dependencies
Training

Learn latent CCG that recovers SRL

• Optimize marginal likelihood

He opened the door

ARG0  ARG1

He opened the door
Features

• **Supertagging features**
  
  • e.g. `open:(S\NP)/NP`

• **Labelling features**
  
  • `open:(S|NP_{ARG0})/NP`
  
  • `open:S|NP_{ARG1}`

• **Attachment features**
  
  • e.g. `open_{ARG0}=door`
This talk

1. Joint model for CCG parsing and SRL

2. Efficient A* parsing algorithm
A* Parsing

- **CKY** chart parsing builds every possible parse
- **Shift-reduce** parsing builds single parse, which may be suboptimal
- **A** searches for single optimal parse
A* Search

Optimistic estimate of distance to goal

Constraints

Agenda item

Cost so far

Goal
A* Parsing

CCG Grammar

Complete parse

Bound on outside score

Partial parse

Inside score
A* Parsing

- Agenda of entries to add to the chart
- Agenda sorted by cost:
  
  \[ cost = \text{inside score} + \text{upper bound on outside score} \]
- First entry spanning chart is guaranteed to be optimal
A* Parsing

• Potentially very efficient

• In practice, overheads mean often slower than CKY
Supertag-factored A* Parsing

Fruit flies like bananas

NP S\NP (S\S)/NP NP

S\S

S\NP

S

Fruit flies like bananas

NP/\NP NP (S\NP)/NP NP

NP

S\NP

S
Supertag-factored A* Parsing

\[
f \left( \begin{array}{c}
\text{Fruit flies like bananas} \\
\text{NP/NP} & \text{NP} & (S\text{NP})/\text{NP} & \text{NP} \\
\text{NP} & \text{S\text{NP}} \\
\text{s} \end{array} \right)
\]

\[
= f(\text{Fruit})_{\text{NP/NP}} + f(\text{flies})_{\text{NP}} + f(\text{like})_{(S\text{NP})/\text{NP}} + f(\text{bananas})_{\text{NP}}
\]
Supertag-factored A* Parsing

\[
f \left( \begin{array}{c}
\text{Fruit flies like bananas} \\
\text{NP} \\
\text{NP} \\
\text{NP} \\
\text{s} \\
\text{(S\textbackslash S)/NP} \\
\text{NP} \\
\text{s} \\
\text{s} \\
\text{s}
\end{array} \right)
\]

\[
= f(\text{Fruit}_{NP}) + f(\text{flies}_{S\textbackslash NP}) + f(\text{like}_{(S\textbackslash S)/NP}) + f(\text{bananas}_{NP})
\]
Supertag-factored A* Parsing

\[
\text{Fruit \ flies like bananas}
\]

\[
\begin{align*}
\text{inside score} &= f\left(\frac{\text{like}}{(S\backslash NP)/NP}\right) + f\left(\frac{\text{bananas}}{NP}\right) \\
\text{outside score} &\leq \max_{\text{cat}} f\left(\frac{\text{Fruit}}{\text{cat}}\right) + \max_{\text{cat}} f\left(\frac{\text{flies}}{\text{cat}}\right)
\end{align*}
\]
Joint A* CCG/SRL Parsing

1. Factorize score for parse into score for words
2. Then compute upper bounds based on sum of score of words

*How do we generalize to a model with dependency features?*
Joint A* CCG/SRL Parsing

Fruit flies like bananas.

Diagram:

```
Fruit  flies  like  bananas
  ?     ?    (S\NP)/NP   NP
  ?     A?    A1
```

```markdown
S\NP
```
Factorization

\[
f \left( \begin{array}{c}
\text{Fruit flies like bananas} \\
\text{NP/NP} & \text{NP} & \text{(S\text{NP})/NP} & \text{NP} \\
\text{NP} & \text{S\text{NP}} & \text{S} \\
\end{array} \right)
\]

\[
= f(\text{Fruit}_{\text{NP/NP}}) + f(\text{flies}_{\text{NP}}) + f(\text{like}_{(\text{S\text{NP})/NP}}) + f(\text{bananas}_{\text{NP}})
\]
Factorization

\[ f \left( \begin{array}{c}
\text{Fruit} \quad \text{flies} \\
\text{NP/NP} \quad \text{NP} \\
\text{NP} \\
\text{S/NP}
\end{array} \right) = f(\text{Fruit}) + f(\text{flies}) + f(\text{like}) + f(\text{bananas}) \]
Factorization

\[ f(\text{Fruit flies like bananas}) = f(\text{Fruit NP}) + f(\text{flies S\NP}) + f(\text{like (S\S)/NP}) + f(\text{bananas NP}) \]
Upper Bounds

• Score for parse is sum of scores for lexical entries

• Upper bound for parse is sum of upper bounds for lexical entries
Upper Bounds for Words

Explicitly computing $\max$ over all lexical entries for a word is intractable

flies
NP
flies
S\NP
A0=fruit
flies
S\NP
A1=fruit
flies
(S\NP)/NP
A0=fruit
A1=bananas
Upper Bounds for Words

flies
NP
flies
S\NP
A0=fruit

flies
S\NP
(S\NP)/NP
A0=fruit
A1=bananas

flies
NP
flies
S\NP
(S\NP)/NP

ARG0
Fruit
ARG1
flies
like
bananas
NONE

Fruit
flies
like
bananas
Upper Bounds for Words

flies
NP

flies
S\NP
A0=

flies
S\NP
A1=

flies
(S\NP)/NP

NP
S\NP
(S\NP)

flies
ARG0
A0=
Fruit

flies
ARG1
A1=
flies
like

bananas

NONE

Fruit
flies
like
bananas
Upper Bounds for Words

flies NP

flies S\NP
A0=fruit

flies S\NP

flies S\NP

A0=
A1=

flies S\NP

flies S\NP

A0=
A1=

flies S\NP

flies S\NP

flies S\NP

NP

(S\NP)/NP

(S\NP)/NP

(S\NP)/NP

NP

(S\NP)/NP

(S\NP)/NP

ARG0

ARG1

NONE

Fruit

flies
like
bananas
Upper Bounds for Words
Lexical Factorization

category features

labelling features

attachment features

flies

S\NP

NP

S\NP

(S\NP)/NP

ARG0

ARG1

NONE

Fruit

flies

like

bananas
Computing Upper bounds

• Score for parse decomposes over lexical entries

\[
f \left( \frac{\text{Fruit flies like bananas}}{\text{NP/NP}} \right) = f(\text{NP/NP}) + f(\text{flies NP}) + f(\text{like (NP)/NP}) + f(\text{bananas NP})
\]
Computing Upper bounds

- Score for parse decomposes over lexical entries
- Lexical entries generated by CFG

\[
\text{flies} \in (S \backslash NP)/NP
\]

A0 = fruit \quad A1 = bananas

\[
\text{flies} \in \text{NP} \quad \text{SINP} \quad (S \backslash NP)/NP
\]

ARG0 \quad ARG1 \quad NONE

Fruit \quad flies \quad like \quad bananas
Computing Upper bounds

- Score for parse decomposes over lexical entries
- Lexical entries generated by CFG
- Features decomposes over rules of CFG
Computing Upper bounds

- Score for parse decomposes over lexical entries
- Lexical entries generated by CFG
- Features decompose over rules of CFG
- Compute upper bounds for lexical entry using Viterbi
Updating Upper Bounds

Fruit flies like bananas

Updating Upper Bounds

Fruit flies like bananas

(S\text{NP})/\text{NP}
Updating Upper Bounds

Fruit flies like bananas
Fruit flies like bananas.
A* Summary

• Factor parsing model into lexical entries

• Generate lexical entries with CFG

• Compute upper bounds for (partially specified) lexical entries using Viterbi algorithm
Experimental Setup

• Evaluate on PropBank as dependency-based SRL with automatic predicate identification

• Relaxed head-matching to avoid arbitrary headedness decisions
SRL Results

- F1 scores for different methods:
  - Riedel
  - Zhao
  - Che
  - Vickrey
  - Pipeline
  - Joint

The Joint method has the highest F1 score, significantly outperforming the other methods.
Out-of-domain SRL Results

F1

- Riedel
- Zhao
- Che
- Vickrey
- Pipeline
- Joint
Efficiency Experiments

![Bar Chart]

- **F1**
  - CKY: 83.5
  - Adaptive Supertagging: 83.1
  - A*: 83.4

- **Speed**
  - CKY: 30 sentences/second
  - Adaptive Supertagging: 7 sentences/second
  - A*: 35 sentences/second
CCG Parsing Results

F1

- Pipeline
- Joint
Conclusions

• First state-of-the-art joint model for syntax and semantics
• First A* parsing algorithm with dependencies to outperform CKY
• Step towards wide-coverage logical forms
Thanks!

Any questions?

https://github.com/mikelewis0/EasySRL