Greedy, Joint Syntactic-Semantic Parsing with Stack LSTMs

Swabha Swayamdipta¹  Miguel Ballesteros²
Chris Dyer³  Noah A. Smith⁴

¹School of Computer Science, Carnegie Mellon University
Pittsburgh, USA

²NLP Group, Universitat Pompeu Fabra
Barcelona, Spain

³Google DeepMind
London, UK

⁴Computer Science and Engineering, University of Washington
Seattle, USA

12th Aug
(CoNLL 2016)
Joint syntactic-semantic parsing
YM-style Syntactic dependency parsing + PropBank-style semantic role labeling
Joint syntactic-semantic parsing
YM-style Syntactic dependency parsing + PropBank-style semantic role labeling

all are expected to reopen soon
Joint syntactic-semantic parsing

YM-style Syntactic dependency parsing + PropBank-style semantic role labeling

all are expected to reopen soon
Joint syntactic-semantic parsing
YM-style Syntactic dependency parsing + PropBank-style semantic role labeling

all are expected to reopen soon

expect.01 reopen.01
Joint syntactic-semantic parsing
YM-style Syntactic dependency parsing + PropBank-style semantic role labeling

Correspondence between syntactic and semantic dependencies [Levin and Hovav, 1996]
Joint syntactic-semantic parsing
YM-style Syntactic dependency parsing + PropBank-style semantic role labeling

Correspondence between syntactic and semantic dependencies [Levin and Hovav, 1996]

Language understanding: QA, relation extraction, text categorization
A little more about PropBank SRL
[Palmer et al., 2005]
A little more about PropBank SRL
[Palmer et al., 2005]

- Most common solution: pipeline syntax and semantics
A little more about PropBank SRL
[Palmer et al., 2005]

- Most common solution: pipeline syntax and semantics
- Pipelines involve *expensive* feature extraction step
[Johansson, 2009, He et al., 2013]
Most common solution: pipeline syntax and semantics

Pipelines involve *expensive* feature extraction step

[Johansson, 2009, He et al., 2013]

Our approach: incremental, joint parsing of syntax and semantics
A little more about PropBank SRL
[Palmer et al., 2005]

- Most common solution: pipeline syntax and semantics
- Pipelines involve *expensive* feature extraction step
  [Johansson, 2009, He et al., 2013]
- Our approach: incremental, joint parsing of syntax and semantics

Pipelines

- Have access to complete syntactic information
- Incremental, joint approach
  - No such access
A little more about PropBank SRL
[Palmer et al., 2005]

- Most common solution: pipeline syntax and semantics
- Pipelines involve *expensive* feature extraction step
  [Johansson, 2009, He et al., 2013]
- Our approach: incremental, joint parsing of syntax and semantics

### Pipelines
- Have access to complete syntactic information
- Slow feature extraction step

### Incremental, joint approach
- No such access
- Fast
Incremental algorithm

- Parse structure → sequence of transitions

Modified arc-eager algorithm [Nivre, 2008, Henderson et al., 2008, Henderson et al., 2013, Gesmundo et al., 2009, Titov et al., 2009]
Incremental algorithm

- Parse structure $\rightarrow$ sequence of transitions
- Transition: \textbf{shift} and \textbf{reduce} actions
Incremental algorithm

- Parse structure → sequence of transitions
- Transition: **shift** and **reduce** actions
- Data structures: **stack** and **buffer**
Incremental algorithm

- Parse structure $\rightarrow$ sequence of transitions
- Transition: **shift** and **reduce** actions
- Data structures: **stack** and **buffer**
- Initialize the **stack** as empty and the **buffer** to contain the sentence
Incremental algorithm

- Parse structure → sequence of transitions
- Transition: **shift** and **reduce** actions
- Data structures: *stack* and *buffer*
- Initialize the *stack* as empty and the *buffer* to contain the sentence
- At each time step, track:
  - Data structure contents (*parser state*)
  - History of transitions

Modified arc-eager algorithm \[Nivre, 2008, Henderson et al., 2008, Henderson et al., 2013, Gesmundo et al., 2009, Titov et al., 2009\]
Incremental algorithm

- Parse structure $\rightarrow$ sequence of transitions
- Transition: **shift** and **reduce** actions
- Data structures: **stack** and **buffer**
- Initialize the **stack** as empty and the **buffer** to contain the sentence
- At each time step, track:
  - Data structure contents (**parser state**)
  - History of transitions
- Terminate when the **buffer** is empty
Incremental algorithm

- Parse structure → sequence of transitions
- Transition: **shift** and **reduce** actions
- Data structures: **stack** and **buffer**
- Initialize the **stack** as empty and the **buffer** to contain the sentence
- At each time step, track:
  - Data structure contents (**parser state**)
  - History of transitions
- Terminate when the **buffer** is empty

Modified arc-eager algorithm [Nivre, 2008, Henderson et al., 2008, Henderson et al., 2013, Gesmundo et al., 2009, Titov et al., 2009]
Transitions for syntax
Transitions for syntax

all are expected to reopen soon

Stack

Buffer

▶ S-Shift

$
Transitions for syntax

all are expected to reopen soon

- S-Shift ✓
- ▶
- ▶
- ▶

Stack

Buffer

all

| are | expected | to | reopen | soon | $ |
Transitions for syntax

all are expected to reopen soon

Stack

Buffer

- S-Shift
- S-Left

[$]
Transitions for syntax

all are expected to reopen soon

Stack

Buffer

- S-Shift
- S-Left ✓
Transitions for syntax

all are expected to reopen soon

[are] expected to reopen soon

S-Shift
S-Left ✓
Transitions for syntax

-all are expected to reopen soon-

▶ S-Shift
▶ S-Left
▶ S-Right

Stack

Buffer

- reopen
- [to]
- [expected]
- [are]

soon $
Transitions for syntax

all are expected to reopen soon

Stack

Buffer

soon

reopen $^{tmp}$ soon
[to]
[expected]
[are]

$
Transitions for syntax

all are expected to reopen soon

Stack

Buffer

soon
[reopen]
[to]
[expected]
[are]

$
Transitions for syntax

- S-Shift
- S-Left
- S-Right
- S-Reduce ✓

Stack

Buffer

all are expected to reopen soon

[reopen]
[to]
[expected]
[are]

$
More transitions for semantics

- M-Shift
- M-Left
- M-Right
- M-Reduce
More transitions for semantics

we make and break agreements

Stack

 Buffer

$\text{Buffer}$

M-Pred

[make]

we

and

$\text{make.03}$

break agreements $\$$
More transitions for semantics

we make and break agreements

Stack

[make]

we

Buffer

break agreements

$
More transitions for semantics

we make and break agreements

Stack

[make]

we

Buffer

[break]

agreements

$
More transitions for semantics

we make and break agreements

[make] break.

Stack

Buffer

M-Pred

M-Swap ✓
More transitions for semantics

- make
- break
- agreements

Stack

Buffer

- M-Pred
- M-Swap
- M-Self
More transitions for semantics

We make and break agreements.

Stack

Buffer

- M-Pred
- M-Swap
- M-Self ✓
Synchronizing syntax and semantics

- Two stacks: *Syn-Stack* and *Sem-Stack*
- Share the *Buffer*
- Syntactic transitions followed by semantic transitions for a given *Buffer* state [Henderson et al., 2008]
An example transition sequence

all are expected to reopen soon

[ x ] denotes parse substructure headed by x
An example transition sequence

all are expected to reopen soon

History
S-Shift

[ x ] denotes parse substructure headed by x
An example transition sequence

all are expected to reopen soon

Syn-Stack

Sem-Stack

Buffer

History

S-Shift
M-Shift

[ x ] denotes parse substructure headed by x
An example transition sequence

all are expected to reopen soon

History

S-Shift
M-Shift
S-Left(sbj)

Syn-Stack

Sem-Stack

Buffer

[×] denotes parse substructure headed by ×
An example transition sequence

all are expected to reopen soon

History
S-Shift
M-Shift
S-Left(sbj)
S-Shift

Syn-Stack

Sem-Stack

Buffer

[ x ] denotes parse substructure headed by x
An example transition sequence

all are expected to reopen soon

History

... M-Shift S-Left(sbj) S-Shift M-Shift

Syn-Stack

Sem-Stack

Buffer

[ × ] denotes parse substructure headed by x
An example transition sequence

text: all are expected to reopen soon

Parsing symbols:
- *are* [Red]: Syn-Stack
- *expected* [Red]: Syn-Stack
- *all* [Blue]: Sem-Stack
- *expected* [Blue]: Sem-Stack
- *soon* [Blue]: Buffer
- *$* [Blue]: Buffer

Symbols:
- `S-Left(sbj)`
- `S-Shift`
- `M-Shift`
- `S-Right(vc)`

`[x]` denotes parse substructure headed by `x`
An example transition sequence

all are expected to reopen soon

**History**

...  
S-Shift  
M-Shift  
S-Right(vc)  
M-Pred(expect.01)

**Buffer**

[expected]  
...  
soon  
$

**Syn-Stack**

[are]  
all

**Sem-Stack**

expected  
[are]  
[are]  

[ × ] denotes parse substructure headed by x
An example transition sequence

**Syn-Stack**

- expected
- [are]

**Sem-Stack**

- all

**Buffer**

- [expected]
- \(...\)
- \(\text{soon}\)
- \(\$\)

**History**

- M-Shift
- S-Right(vc)
- M-Pred(expect.01)
- M-Reduce

\(\text{all are expected to reopen soon}\)

\(\text{expect.01}\)
An example transition sequence

**Syn-Stack**
- **expected**
- **[are]**

**Sem-Stack**
- **all**

**Buffer**
- \([\text{expected}]\)
- \(\ldots\)
- **soon**
- $\$

**History**

- S-Right(vc)
- M-Pred(expect.01)
- M-Reduce
- M-Left(A1)

\([x]\) denotes parse substructure headed by \(x\)
An example transition sequence

all are expected to reopen soon

History

... M-Pred(expect.01)
M-Reduce
M-Left(A1)
M-Shift

[ x ] denotes parse substructure headed by x
An example transition sequence

**History**

```
...  
M-Reduce
M-Left(A1)
M-Shift
```

**Syn-Stack**

```
[are]
expected
all
```

**Sem-Stack**

```
[expected]
```

**Buffer**

```
to  reopen  soon  $
```

[ x ] denotes parse substructure headed by x
An example transition sequence

all are expected to reopen soon

History

... M-Left(A1) M-Shift S-Right(oprd)

Syn-Stack

to [expected] [are] all

Sem-Stack

[expected] to reopen soon

Buffer

[ x ] denotes parse substructure headed by x
An example transition sequence

all are expected to reopen soon

History

... M-Shift
S-Right(oprd)
M-Right(C-A1)

[ ] denotes parse substructure headed by x
An example transition sequence

all are expected to reopen soon

Sbj
vc
opr
A1
C-A1

History

... 
S-Right(oprd)
M-Right(C-A1)
M-Reduce

Syn-Stack

to
[expected]
[are]

all

Sem-Stack

to
reopen
soon
$

Buffer

[ x ] denotes parse substructure headed by x
An example transition sequence

all are expected to reopen soon

[×] denotes parse substructure headed by x
An example transition sequence

all are expected to reopen soon

History

... 
M-Reduce 
M-Shift 
S-Right(im)

[ x ] denotes parse substructure headed by x
An example transition sequence

**History**

...  
M-Shift  
S-Right(im)  
M-Pred(reopen.01)

**Syn-Stack**

- reopen
- [to]
- [expected]
- [are]

**Sem-Stack**

- to
- all

**Buffer**

- [reopen]
- soon
- $
An example transition sequence

all are expected to reopen soon

History

…
S-Right(im)
M-Pred(reopen.01)
M-Reduce

[ x ] denotes parse substructure headed by x
An example transition sequence

**History**

...  
M-Pred(reopen.01)  
M-Reduce  
M-Left(A1)

```
[reopen]  soon  $
```

**Syn-Stack**

```
reopen  [to]  [expected]  [are]  all
```

**Buffer**

```
[reopen]  soon  $
```

[x] denotes parse substructure headed by x
An example transition sequence

all are expected to reopen soon

History

... M-Reduce
M-Left(A1) M-Reduce

[ x ] denotes parse substructure headed by x
An example transition sequence

all are expected to reopen soon

History

... M-Left(A1) M-Reduce M-Shift

Syn-Stack

reopen
[to]
[expected]
[are]

Sem-Stack

reopen

Buffer

[ soon ] $
An example transition sequence

all are expected to reopen soon

History

... M-Reduce M-Shift S-Right(tmp)

Syn-Stack

soon [reopen] [to] [expected] [are]

Sem-Stack

reopen

Buffer

soon $ [ x ]$ denotes parse substructure headed by $x$
An example transition sequence

all are expected to reopen soon

Syntactic history:
- M-Shift
- S-Right(tmp)
- M-Right(AM-TMP)

Semantic history:
- [are]
- [expected]
- [to]
- [reopen]
- soon

Buffer:
- soon $
An example transition sequence

```
all are expected to reopen soon

Sbj   Vc     Oprd   Im     Tmp
A1    C-A1   A1    AM-TMP
```

History

```
S-Right(tmp)
M-Right(AM-TMP)
M-Reduce
```

[ x ] denotes parse substructure headed by x
An example transition sequence

```
all are expected to reopen soon
```

**History**

... 
M-Right(AM-TMP)
M-Reduce
M-Shift

[ x ] denotes parse substructure headed by x

**Syn-Stack**

soon
[reopen]
[to]
[expected]
[are]

**Sem-Stack**

soon

$\quad$

**Buffer**

$\quad$
An example transition sequence

all are expected to reopen soon

Syn-Stack
- [reopen]
- [to]
- [expected]
- [are]

Sem-Stack
- soon

History
... M-Reduce M-Shift S-Reduce

Buffer
$ 

[ x ] denotes parse substructure headed by x
An example transition sequence

```
all are expected to reopen soon
```

```
expect.01 reopen.01
```

```
M-Shift
S-Reduce
S-Reduce
```

[to] [expected] [are]

**Syn-Stack**

**Sem-Stack**

Buffer

9/25
An example transition sequence

all are expected to reopen soon

expect.01 reopen.01

[expected] [are]

Syn-Stack

soon

Sem-Stack

Buffer

[ x ] denotes parse substructure headed by x

History

... S-Reduce S-Reduce S-Reduce
An example transition sequence

all are expected to reopen soon

expect.01 reopen.01

[ are ]

Syn-Stack

soon

Sem-Stack

$\text{Buffer}$

[ $\times$ ] denotes parse substructure headed by $\times$
An example transition sequence

all are expected to reopen soon

expect.01 reopen.01

Syn-Stack

Sem-Stack

Buffer

History

... S-Reduce S-Reduce S-Left($)

$ [ $ $ ]

{x} denotes parse substructure headed by x
An example transition sequence

**History**

...  
S-Reduce  
S-Left($)  
S-Shift

[ [ x ] denotes parse substructure headed by x ]
An example transition sequence

all are expected to reopen soon

Syn-Stack

Sem-Stack

Buffer

$ [ x ]$ denotes parse substructure headed by $x$

History

S-Left$(\$)$
S-Shift
M-Reduce
An example transition sequence

```
all are expected to reopen soon
```

Linear algorithm

```
Syn-Stack

Sem-Stack

Buffer
```

History

```
S-Left($)
S-Shift
M-Reduce
M-Shift
```
Outline

Introduction

Incremental Algorithm

Stack LSTM Model

CoNLL 2008-09 Shared Task Results
Stack LSTM Model

- **LSTMs**: Recurrent neural networks with special memory cell [Hochreiter and Schmidhuber, 1997, Graves, 2013] to learn fixed-size representations for variable-length sequences

- **Stack LSTMs**: LSTMs equipped with stack operations [Dyer et al., 2015]
  - summary → return a fixed-size continuous representation
  - push → add to the sequence
  - pop → remove from the sequence
Stack LSTM for Joint Parsing

History

...  
M-Reduce  
M-Left(A1)  
M-Shift

Buffer

[to] reopen soon $  

[×] denotes parse substructure headed by x

to [expected]  
[are] all

Syn-Stack  
Sem-Stack
Stack LSTM for Joint Parsing

S-Right (oprd)

History

... M-Reduce M-Left(A1) M-Shift

[×] denotes parse substructure headed by x

Syn-Stack

Sem-Stack

to [expected] [are]

Buffer

to reopen soon $
Stack LSTM for Joint Parsing

\[ \text{S-Right (oprbd)} \]

\begin{align*}
\text{Syn-Stack} & \quad \text{Sem-Stack} \\
to & \quad [\text{expected}] & [\text{are}] \\
[\text{expected}] & \quad \text{all} \\
\end{align*}

\[ \text{History} \]

\[
\ldots
\]

M-Reduce
M-Left(A1)
M-Shift

\[ \text{Greedy} \]

\[ \text{Buffer} \]

\[
\begin{array}{c}
to \\
reopen \\
soon \\
$ \\
\end{array}
\]

[ \times ] denotes parse substructure headed by x
Outline

Introduction

Incremental Algorithm

Stack LSTM Model

CoNLL 2008-09 Shared Task Results
CoNLL Shared Tasks

- 2008: English only
- 2009: Multilingual
- Evaluation metrics:
  - Syntax: Labeled Accuracy Score (LAS)
  - SRL: Semantic $F_1$
  - Rank systems: Macro $F_1$
Experimental Setup

- POS tags were used, but no other provided features
- No manually-designed features
- Dataset-specific hyperparameter tuning
CoNLL 2008 (English only) Shared Task

Macro $F_1$

Models

- Lluis:08
- Henderson:08
- Johansson:09
- Johansson:09
- Titov:09
- Zhao:08
- Che:08
- Ciaramita:08
- Our
- Our
CoNLL 2009 (Multilingual) Shared Task

![Graph showing Macro F1 scores for different languages and models.](image-url)

- **Languages:** En (English), Ja (Japanese), Sp (Spanish), De (German), Ca (Catalan), Cz (Czech), Ch (Chinese)
- **Models:** "[Che et al., 2009]", "[Zhao et al., 2009]", "[Gesmundo et al., 2009]", "Our"
- **Scores:**
  - En: 86
  - Ja: 82
  - Sp: 82
  - De: 82
  - Ca: 82
  - Cz: 82
  - Ch: 76
Conclusion

Take-aways!

- Incremental algorithm (linear) + model using stack LSTMs
- Avoid the effort involved in manual feature engineering
- Light-weight alternative to expensive pipelined systems

Code available at
https://github.com/clab/joint-lstm-parser


Syntactic-semantic composition
SRL performance on out-of-domain (Brown) data
CoNLL 2009 Shared Task

Models

Semantic $F_1$

Gesmundo:09
Che:09
Bjorkelund:10
Our
Zhao:09
Tackstrom:15
Fitzgerald:15
Lei:15
Roth:14
Roth:16
Time to decode the CoNLL 2009 English dataset

Experiments were run end to end on a single CPU

[Lei et al., 2015]