Joint Parsing and Disfluency Detection in Linear Time

Mohammad Sadegh Rasooli  Joel Tetreault

Columbia University

Nuance Communications
Speech Disfluency

- Speech text is mostly disfluent
- Disfluency types:
  - Filled pauses; e.g. *uh, um*
  - Discourse markers and parentheticals; e.g. *I mean, you know*
  - Reparandum (edited phrase)

I want a flight to Boston *uh I mean to Denver*
Most prior approaches focus solely on disfluency detection.

Why not parse the disfluent sentence at the same time as disfluency detection?

✓ This has the potential to speed-up spoken language processing in dialogue systems.
Parsingspoken language is harder than written text. Disfluencies make it much harder.

How about joint parsing?
Studies that only focus on disfluency detection vastly outperform joint model approaches by 20 F score or more.
Our Approach: Joint Parsing and Disfluency Detection

- Parsing and disfluency detection with high accuracy and processing speed.

  I want a flight to Boston uh I mean to Denver
  I want a flight to Denver

This is the real output of our system!
Our Approach: Joint Parsing and Disfluency Detection

- Parsing and disfluency detection with high accuracy and processing speed.
  
  I want a flight to Boston uh I mean to Denver
  I want a flight to Denver

This is the real output of our system!
Introduction
Speech Disfluency
Parsing Disfluent Sentences

Joint Dependency Parsing and Disfluency Detection
Arc-Eager Parsing
Additional Transitions for Handling Disfluencies
Learning Model

Evaluation
Disfluency Detection
Parser Evaluation
Arc-Eager Parsing [Nivre, 2004]

- **Goal**: Finding the best dependency tree
- **Parser State**: Buffer of words, stack of already processed words and set of already made dependency arcs.
- **Initialization**: Buffer with sentence words, stack and arc-set are empty.
- **Final State**: Stack and buffer are empty and arc-set has a set of arcs.
Arc-Eager Parsing

Actions in an arc-eager algorithms are:

- **Shift**: $[... \ j \]_S [i \ k \ ...]_B \rightarrow [... \ j \ i \]_S [k \ ...]_B$
- **Right-arc**: $[... \ j \]_S [i \ k \ ...]_B \rightarrow [... \ j \ i \]_S [k \ ...]_B + \text{add-arc}(j,i)$
- **Left-arc**: $[... \ h \ j \]_S [i \ k \ ...]_B \rightarrow [...h \]_S [i \ k \ ...]_B + \text{add-arc}(i,j)$
- **Reduce**: $[... \ h \ j \]_S [i \ k \ ...]_B \rightarrow [...h \]_S [i \ k \ ...]_B$
- Are these actions ENOUGH for disfluency detection?
Arc-Eager Parsing

Actions in an arc-eager algorithms are:

- **Shift**: \([... \ j \ S \ [i \ k \ ...]_B \rightarrow [... \ j \ i \ S \ [k \ ...]_B\)
- **Right-arc**: \([... \ j \ S \ [i \ k \ ...]_B \rightarrow [... \ j \ i \ S \ [k \ ...]_B + \text{add-arc}(j,i)\)
- **Left-arc**: \([... \ h \ j \ S \ [i \ k \ ...]_B \rightarrow [...h \ S \ [i \ k \ ...]_B + \text{add-arc}(i,j)\)
- **Reduce**: \([... \ h \ j \ S \ [i \ k \ ...]_B \rightarrow [...h \ S \ [i \ k \ ...]_B\)

- **Are these actions ENOUGH for disfluency detection?**
Additional Transitions for Handling Disfluencies

- Three additional actions:
  
  \( \text{Intj}[i] \): Remove the first \( i \) words from the buffer and tag them as \textit{interjection} (\text{Intj}).

\[
\begin{align*}
\text{ROOT}_0, \text{want}_2, \text{flight}_4, \text{to}_5, \text{Boston}_6]_S & \quad \text{[uh}_7, \text{l}_8, \text{mean}_9, \text{to}_10, \text{Denver}_11]_B \quad \rightarrow \text{Next action is Intj}[1] \\
\text{ROOT}_0, \text{want}_2, \text{flight}_4, \text{to}_5, \text{Boston}_6]_S & \quad \text{[l}_8, \text{mean}_9, \text{to}_10, \text{Denver}_11]_B
\end{align*}
\]
Three additional actions:

\textbf{Intj}[i]: Remove the first $i$ words from the buffer and tag them as \textit{interjection} (Intj).

\begin{align*}
[\text{ROOT}_0, \text{want}_2, \text{flight}_4, \text{to}_5, \text{Boston}_6]_S & \quad \rightarrow \quad \text{Next action is } \text{Intj}[1] \\
[\text{ROOT}_0, \text{want}_2, \text{flight}_4, \text{to}_5, \text{Boston}_6]_S & \quad \rightarrow \quad \text{Intj}[1]
\end{align*}
Additional Transitions for Handling Disfluencies

- Three additional actions:
  \[ \text{Intj}[i] \]: Remove the first \( i \) words from the buffer and tag them as *interjection* (Intj).

\[
\begin{align*}
[\text{ROOT}_0, \text{want}_2, \text{flight}_4, \text{to}_5, \text{Boston}_6]_S & \quad [\text{uh}_7, \text{I}_8, \text{mean}_9, \text{to}_10, \text{Denver}_{11}]_B \\
\rightarrow \text{Next action is Intj}[1] & \quad [\text{ROOT}_0, \text{want}_2, \text{flight}_4, \text{to}_5, \text{Boston}_6]_S \\
& \quad [\text{I}_8, \text{mean}_9, \text{to}_10, \text{Denver}_{11}]_B
\end{align*}
\]
Three additional actions:

- \text{Intj}[i]$: Remove the first $i$ words from the buffer and tag them as \textit{interjection} (\text{Intj}).

- [ROOT$_0$, want$_2$, flight$_4$, to$_5$, Boston$_6$]$_S$  $\rightarrow$ Next action is \text{Intj}[1]  
- [ROOT$_0$, want$_2$, flight$_4$, to$_5$, Boston$_6$]$_S$  
- [uh$_7$, I$_8$, mean$_9$, to$_{10}$, Denver$_{11}$]$_B$  
- [I$_8$, mean$_9$, to$_{10}$, Denver$_{11}$]$_B$
Three additional actions:

\[ \text{Prn}[i] \]: Remove the first \( i \) words from the buffer and tag them as \textit{discourse marker} (Prn).

\[ \text{ROOT}_0, \text{want}_2, \text{flight}_4, \text{to}_5, \text{Boston}_6 \] \_S \ 
\rightarrow \text{Next action is Prn}[2]
\[ \text{ROOT}_0, \text{want}_2, \text{flight}_4, \text{to}_5, \text{Boston}_6 \] \_S \ 
\[ \text{l}_8, \text{mean}_9, \text{to}_10, \text{Denver}_11 \] \_B

\[ \text{to}_10, \text{Denver}_11 \] \_B
Three additional actions:

- \textbf{Prn}[i]: Remove the first \( i \) words from the buffer and tag them as \textit{discourse marker} (\textbf{Prn}).

\[
\text{[ROOT}_0, \text{want}_2, \text{flight}_4, \text{to}_5, \text{Boston}_6]_S \quad \text{[l}_8, \text{mean}_9, \text{to}_10, \text{Denver}_{11}]_B
\]

\rightarrow \text{Next action is Prn}[2]

\[
\text{[ROOT}_0, \text{want}_2, \text{flight}_4, \text{to}_5, \text{Boston}_6]_S \quad \text{[to}_10, \text{Denver}_{11}]_B
\]
Additional Transitions for Handling Disfluencies

- Three additional actions:
  - \( \text{Prn}[i] \): Remove the first \( i \) words from the buffer and tag them as *discourse marker* (\( \text{Prn} \)).

\[
\text{[ROOT}_0, \text{want}_2, \text{flight}_4, \text{to}_5, \text{Boston}_6]_S \rightarrow \text{Next action is Prn}[2] \Rightarrow \text{[ROOT}_0, \text{want}_2, \text{flight}_4, \text{to}_5, \text{Boston}_6]_S
\]

\[
\text{[I}_8, \text{mean}_9, \text{to}_10, \text{Denver}_{11}]_B \Rightarrow \text{[to}_10, \text{Denver}_{11}]_B
\]
Three additional actions:

- \textbf{Prn}[i]: Remove the first \( i \) words from the buffer and tag them as \textit{discourse marker} (Prn).

\[ \text{Next action is Prn[2]} \]

\[ [\text{ROOT}_0, \text{want}_2, \text{flight}_4, \text{to}_5, \text{Boston}_6]_S \quad [l_8, \text{mean}_9, \text{to}_10, \text{Denver}_{11}]_B \]

\[ [\text{ROOT}_0, \text{want}_2, \text{flight}_4, \text{to}_5, \text{Boston}_6]_S \quad [\text{to}_10, \text{Denver}_{11}]_B \]
Three additional actions:

- **RP[i:j]**: From the words outside the buffer, remove un-removed words $i$ to $j$ and tag them as *reparandum* (RP).

  \[ \text{Candidates: RP[6:6], RP[5:6], RP[4:6], RP[3:6], ..., Intj[1], Intj[2], ..., Prn[1], Prn[2], ..., Shift, Reduce, Left-arc, Right-arc} \]

→ Next action is **RP[5:6]**

  \[ \text{[ROOT}_0, \text{want}_2, \text{flight}_4]_S \quad \text{[to}_10, \text{Denver}_11]_B \]
Three additional actions:

- **RP[i:j]**: From the words outside the buffer, remove un-removed words $i$ to $j$ and tag them as *reparandum* (RP).

  
  \[ [\text{ROOT}_0, \text{want}_2, \text{flight}_4, \text{to}_5, \text{Boston}_6]_S \quad [\text{to}_{10}, \text{Denver}_{11}]_B \]


  \rightarrow \text{Next action is RP[5:6]}

  \[ [\text{ROOT}_0, \text{want}_2, \text{flight}_4]_S \quad [\text{to}_{10}, \text{Denver}_{11}]_B \]
Three additional actions:

- **RP[i:j]**: From the words outside the buffer, remove un-removed words \(i\) to \(j\) and tag them as *reparandum* (RP).

Consider the sentence: 

\[
[\text{ROOT}_0, \text{want}_2, \text{flight}_4, \text{to}_5, \text{Boston}_{6}]_S \quad [\text{to}_{10}, \text{Denver}_{11}]_B
\]


→ Next action is RP[5:6]

\[
[\text{ROOT}_0, \text{want}_2, \text{flight}_4]_S \quad [\text{to}_{10}, \text{Denver}_{11}]_B
\]
Three additional actions:

- **RP[i:j]**: From the words outside the buffer, remove un-removed words from $i$ to $j$ and tag them as *reparandum* (**RP**).

\[
\text{Candidates: } \text{RP}[6:6], \text{RP}[5:6], \text{RP}[4:6], \text{RP}[3,6], ..., \text{Intj}[1], \text{Intj}[2], ..., \text{Prn}[1], \text{Prn}[2], ..., \text{Shift}, \text{Reduce}, \text{Left-arc}, \text{Right-arc}
\]

→ Next action is **RP[5:6]**

\[
[\text{ROOT}_0, \text{want}_2, \text{flight}_4]_S \quad [\text{to}_{10}, \text{Denver}_{11}]_B
\]
Three additional actions:

✓ **RP[i:j]**: From the words outside the buffer, remove un-removed words \(i\) to \(j\) and tag them as *reparandum* (RP).

\[
[\text{ROOT}_0, \text{want}_2, \text{flight}_4, \text{to}_5, \text{Boston}_6]_S \quad [\text{to}_{10}, \text{Denver}_{11}]_B
\]


→ Next action is RP[5:6]

\[
[\text{ROOT}_0, \text{want}_2, \text{flight}_4]_S \quad [\text{to}_{10}, \text{Denver}_{11}]_B
\]
Let’s Practice

\[ \text{[ROOT}_0, \text{want}_2, \text{flight}_4]_S \quad \text{[to}_5, \text{Boston}_6, \text{uh}_7, \text{I}_8, \text{mean}_9, \text{to}_{10}, \text{Denver}_{11}]_B \]

Next action is right-arc:prep

\[ \text{[Root]} \quad \text{I want a flight to Boston uh I mean to Denver} \]
Let’s Practice

\[
[\text{ROOT}_0, \text{want}_2, \text{flight}_4, \text{to}_5]_S \quad [\text{Boston}_6, \text{uh}_7, \text{I}_8, \text{mean}_9, \text{to}_10, \text{Denver}_11]_B
\]

Next action is \textbf{right-arc:pobj}

[\textit{Root}] \ I \ want \ a \ flight \ to \ Boston \ uh \ I \ mean \ to \ Denver
Let’s Practice

$[\text{ROOT}_0, \text{want}_2, \text{flight}_4, \text{to}_5, \text{Boston}_6]_S$ $[\text{uh}_7, \text{I}_8, \text{mean}_9, \text{to}_10, \text{Denver}_11]_B$

Next action is \textbf{Intj}[1]

$[\text{Root}]$ I want a flight to Boston uh I mean to Denver
Let’s Practice

\[ \text{[ROOT} \_0, \text{want} \_2, \text{flight} \_4, \text{to} \_5, \text{Boston} \_6]_S \quad [l\_8, \text{mean} \_9, \text{to} \_10, \text{Denver} \_11]_B \]

Next action is **Prn[2]**
[ROOT$_0$, want$_2$, flight$_4$, to$_5$, Boston$_6$]$_S$  [to$_{10}$, Denver$_{11}$]$_B$

Next action is **RP[5:6]**

[Root] I want a flight to Boston to Denver
Let’s Practice

\[ \text{ROOT}_0, \text{want}_2, \text{flight}_4, \text{to}_5, \text{Boston}_6 \]_S \quad [\text{to}_10, \text{Denver}_{11}]_B

Deleting words and dependencies

[Root] I want a flight to Boston to Denver
Let’s Practice

\[ \text{Next action is right-arc:prep} \]

\[ [\text{ROOT}_0, \text{want}_2, \text{flight}_4]_S \quad [\text{to}_10, \text{Denver}_{11}]_B \]

\[ [\text{Root}] \quad \text{I want a flight to Denver} \]
Let’s Practice

\[
[\text{ROOT}_0, \text{want}_2, \text{flight}_4, \text{to}_{10}]_S \quad [\text{Denver}_{11}]_B
\]

Next action is right-arc:pobj

\[
[\text{Root}] \quad \text{I want a flight to Denver}
\]
Two Classifiers for Learning the Model

- Instead of having one complete joint model, we have two classifiers that each classifier has its own features and label set.

![Diagram showing two classifiers C1 and C2 with states and transitions]

- \( C1 \) with states: DM[i], Parse, RP[i:j], IJ[i]
- \( C2 \) with states: SH, LA, RA, R
Features

- We use two kinds of features for the first classifier: local and global.

**Global Features**
- First n words inside/outside buffer (n=1:4)
- First n POS i/o buffer (n=1:6)
- Are n words i/o buffer equal? (n=1:4)
- Are n POS i/o buffer equal? (n=1:4)
- n last FG transitions (n=1:5)
- n last transitions (n=1:5)
- n last FG transitions + first POS in the buffer (n=1:5)
- n last transitions + first POS in the buffer (n=1:5)
- (n+m)-gram of m/n POS i/o buffer (n,m=1:4)
- Refined (n+m)-gram of m/n POS i/o buffer (n,m=1:4)
- Are n first words of i/o buffer equal? (n=1:4)
- Are n first POS of i/o buffer equal? (n=1:4)
- Number of common words i/o buffer words (n=1:6)

**Local Features**
- First n words of the candidate phrase (n=1:4)
- First n POS of the candidate phrase (n=1:6)
- Distance between the candidate and first word in the buffer
Learning Algorithm

- We experimented with two learning algorithms [Collins, 2002]:
  - We use averaged Perceptron [Collins, 2002] with mostly binary features (AP).
  - Changing weight updates from 1 to 2 for misclassification of reparandum shows improvement (WAP).
1 Introduction
   Speech Disfluency
   Parsing Disfluent Sentences

2 Joint Dependency Parsing and Disfluency Detection
   Arc-Eager Parsing
   Additional Transitions for Handling Disfluencies
   Learning Model

3 Evaluation
   Disfluency Detection
   Parser Evaluation
Evaluation Data and Measures

- **Data**
  - ✓ We use Switchboard parsed section (mrg files) with the same train/dev/test split as [Johnson and Charniak, 2004]

- **Metric**
  - ✓ **Disfluency detection**: F-score of detecting reparandum.
  - ✓ **Parsing**: F-score of finding correct parents for fluent words.
### Disfluency Detection

<table>
<thead>
<tr>
<th>Model</th>
<th>Model Description</th>
<th>F-Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>[Miller and Schuler, 2008]</td>
<td>Joint + PCFG parsing</td>
<td>30.6</td>
</tr>
<tr>
<td>[Lease and Johnson, 2006]</td>
<td>Joint + PCFG parsing</td>
<td>62.4</td>
</tr>
<tr>
<td>[Kahn et al., 2005]</td>
<td>TAG + LM rerank.</td>
<td>78.2</td>
</tr>
<tr>
<td>[Qian and Liu, 2013]</td>
<td>IOB tagging</td>
<td>81.4</td>
</tr>
<tr>
<td>[Qian and Liu, 2013]–Opt.</td>
<td>IOB tagging</td>
<td>82.1</td>
</tr>
<tr>
<td>Our Model</td>
<td>AP</td>
<td>80.9</td>
</tr>
<tr>
<td>Our Model</td>
<td>WAP</td>
<td>81.4</td>
</tr>
</tbody>
</table>
Table 1: Parsing results. UB = upperbound (parsing clean sentences), LB = lowerbound (parsing disfluent sentences without disfluency correction). UAS is unlabeled attachment score (accuracy), Pr. is precision, Rec. is recall and F1 is f-score.
Our experiments show that our model is close to the state-of-the-art.

There are still many avenues of improving accuracy:

✓ Better structure: completely joint model
✓ Better features: prosodic features
✓ K-beam training and decoding
✓ Classifier ensemble
Any Question?

Thanks $[\text{for}]_{Rp.}$ $[\text{uh}]_{Intj}$ for your attention
References I


