بسم الله الرحمن الرحیم

Joint Parsing and Disfluency Detection in Linear Time

Mohammad Sadegh Rasooli   Joel Tetreault

Columbia University

Nuance Communications

Overview

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)

$$\text{I want a flight } \underbrace{\text{to Boston}}_{\text{Reparandum}} \; \overbrace{\underbrace{\text{uh}}_{\text{FP}} \; \underbrace{\text{I mean}}_{\text{Prn}}}^{\text{Interregnum}} \; \underbrace{\text{to Denver}}_{\text{Repair}}$$

Parsing Disfluent Sentences

- 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.

Parsing Disfluent Sentences

- Parsing spoken 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

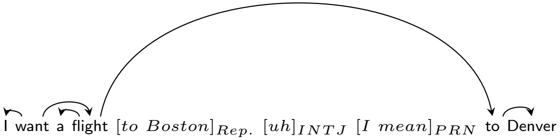
This is the real output of our system!

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!

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 ...]$_B$ + add-arc(j,i)
- **Left-arc:** [... h j]$_S$ [i k ...]$_B$ $\rightarrow$ [...h]$_S$ [i k ...]$_B$ + 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 ...]$_B$ + add-arc(j,i)
- **Left-arc:** [... h j]$_S$ [i k ...]$_B$ $\rightarrow$ [...h]$_S$ [i k ...]$_B$ + 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:

  Intj[i]: Remove the first *i* words from the buffer and tag them as *interjection* (**Intj**).

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

$\rightarrow$ Next action is Intj[1]

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

Additional Transitions for Handling Disfluencies

- Three additional actions:

  Intj[i]: Remove the first *i* words from the buffer and tag them as *interjection* (**Intj**).

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

→ Next action is Intj[1]

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

Additional Transitions for Handling Disfluencies

- Three additional actions:

  Intj[i]: Remove the first *i* words from the buffer and tag them as *interjection* (**Intj**).

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

$\rightarrow$ Next action is Intj[1]

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

Additional Transitions for Handling Disfluencies

- Three additional actions:

  Intj[i]: Remove the first *i* words from the buffer and tag them as *interjection* (**Intj**).

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

$\rightarrow$ Next action is Intj[1]

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

Additional Transitions for Handling Disfluencies

- Three additional actions:

  Prn[i]: Remove the first *i* words from the buffer and tag them as *discourse marker* (**Prn**).

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

$\rightarrow$ Next action is Prn[2]

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

Additional Transitions for Handling Disfluencies

- Three additional actions:

  Prn[i]: Remove the first *i* words from the buffer and tag them as *discourse marker* (**Prn**).

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

$\rightarrow$ 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:

  Prn[i]: Remove the first *i* words from the buffer and tag them as *discourse marker* (**Prn**).

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

$\rightarrow$ Next action is Prn[2]

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

Additional Transitions for Handling Disfluencies

- Three additional actions:

  Prn[i]: Remove the first *i* words from the buffer and tag them as *discourse marker* (**Prn**).

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

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

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

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]

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

Additional Transitions for Handling Disfluencies

- 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**).

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

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]

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

Additional Transitions for Handling Disfluencies

- 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**).

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

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]

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

Additional Transitions for Handling Disfluencies

- 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**).

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

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]

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

Additional Transitions for Handling Disfluencies

- 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**).

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

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

$\rightarrow$ Next action is RP[5:6]

[ROOT$_0$, want$_2$, flight$_4$]$_S$ [to$_{10}$, 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**

Let's Practice

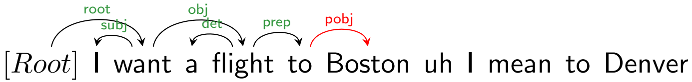
$[\text{ROOT}_0, \text{want}_2, \text{flight}_4, \text{to}_5]_S \quad [\text{Boston}_6, \text{uh}_7, \text{I}_8, \text{mean}_9, \text{to}_{10}, \text{Denver}_{11}]_B$

Next action is **right-arc:pobj**

Let's Practice

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

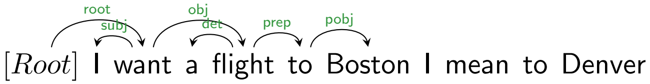
Next action is **Intj[1]**



Let's Practice

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

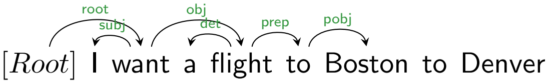
$\boxed{\text{Next action is } \textbf{Prn[2]}}$



Let's Practice

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

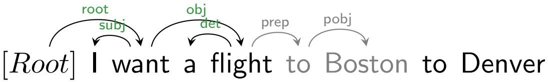
Next action is **RP[5:6]**



Let's Practice

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

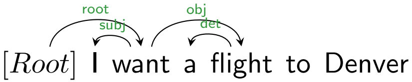
Deleting words and dependencies



Let's Practice

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

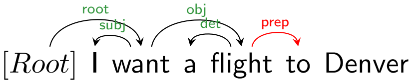
Next action is **right-arc:prep**



Let's Practice

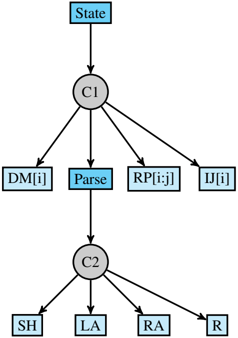
$[\text{ROOT}_0, \text{want}_2, \text{flight}_4, \text{to}_{10}]_S \quad [\text{Denver}_{11}]_B$

Next action is **right-arc:pobj**



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.



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).

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

Model	Model Description	F-Score
[Miller and Schuler, 2008]	Joint + PCFG parsing	30.6
[Lease and Johnson, 2006]	Joint + PCFG parsing	62.4
[Kahn et al., 2005]	TAG + LM rerank.	78.2
[Qian and Liu, 2013]	IOB tagging	81.4
[Qian and Liu, 2013]–Opt.	IOB tagging	82.1
Our Model	AP	80.9
Our Model	WAP	81.4

Parser Evaluation

	UAS	LB	UB	Pr.	Rec.	F1
AP	**88.6**	70.7	90.2	86.8	88.0	87.4
WAP	88.1	70.7	90.2	87.2	88.0	**87.6**

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.

Conclusion and Future Directions

- 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 $[for]_{Rp.}$ $[uh]_{Intj}$ for your attention

References I

Bohnet, B. (2010).
Very high accuracy and fast dependency parsing is not a contradiction.
In *COLING*, pages 89–97.

Collins, M. (2002).
Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms.
In *ACL*, pages 1–8.

Godfrey, J. J., Holliman, E. C., and McDaniel, J. (1992).
Switchboard: Telephone speech corpus for research and development.
In *ICASSP*, volume 1, pages 517–520.

Johnson, M. and Charniak, E. (2004).
A TAG-based noisy channel model of speech repairs.
In *ACL*, pages 33–39.

Kahn, J. G., Lease, M., Charniak, E., Johnson, M., and Ostendorf, M. (2005).
Effective use of prosody in parsing conversational speech.
In *EMNLP*, pages 233–240.

References II

Kübler, S., McDonald, R., and Nivre, J. (2009).
Dependency parsing.
Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Lease, M. and Johnson, M. (2006).
Early deletion of fillers in processing conversational speech.
In *NAACL-HLT*, pages 73–76.

Levy, R. and Andrew, G. (2006).
Tregex and tsurgeon: tools for querying and manipulating tree data structures.
In *LREC*, pages 2231–2234.

Miller, T. and Schuler, W. (2008).
A unified syntactic model for parsing fluent and disfluent speech.
In *ACL-HLT*, pages 105–108.

Nivre, J. (2004).
Incrementality in deterministic dependency parsing.
In *Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*, pages 50–57.

References III

Qian, X. and Liu, Y. (2013).
Disfluency detection using multi-step stacked learning.
In *NAACL-HLT*, pages 820–825.

Zwarts, S. and Johnson, M. (2011).
The impact of language models and loss functions on repair disfluency detection.
In *ACL*, pages 703–711.