# Detecting Speech Repairs Incrementally Using a Noisy Channel Approach

Simon Zwarts, Mark Johnson, Robert Dale

Department of Computing
Macquarie University

COLING 2010

MACQUARIE
UNIVERSITY

# Research goals

- Spontaneous speech often contains *disfluencies*

  *I want a flight to Boston, uh, I mean, to Denver on Friday*

  which we'd like to detect and delete in order to produce a more fluent transcript
- Current disfluency detection/correction systems process entire sentences at a time
- An *incremental speech disfluency detector/corrector* could better integrate with incremental speech recognition
  - and ultimately might not require sentence segmentation
- We describe an incremental version of the Charniak and Johnson (2004) TAG-based model
- We also propose *two new metrics* to measure how quickly and accurately an incremental disfluency system detects disfluencies

# Speech errors in (transcribed) speech

- Filled pauses:

  *I think it's, uh, refreshing to see the, uh, support …*

- Parentheticals:

  *But, you know, I was reading the other day …*

- Speech repairs:

  *Why didn't he, why didn't she stay at home?*

- Ungrammatical constructions:

  *My friends is visiting me?*

# Why focus on speech repairs?

- *Filled pauses* are easy to recognize (in transcripts at least)
- *Parentheticals* are easy to detect (e.g., parsing)
- *"Ungrammatical" constructions* aren't necessarily fatal
  - Statistical parsers learn mapping of sentences to parses in training corpus
- *Speech repairs* warrant special treatment, since standard PCFG-based parsers misanalyse them

# Shriberg's analysis of speech repairs

$$\textit{I want a flight } \underbrace{\textit{to Boston,}}_{\text{reparandum}} \underbrace{\textit{uh, I mean}}_{\text{interregnum}} \underbrace{\textit{to Denver}}_{\text{repair}} \textit{ on Friday}$$

- The Interregnum is usually lexically (and prosodically marked), but can be empty
- Repairs can cross syntactic boundaries

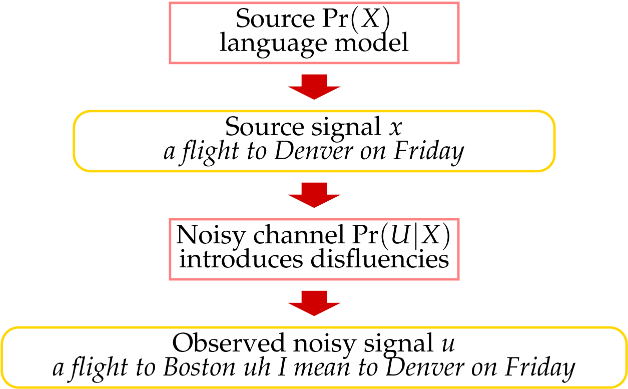
  $\underline{\textit{Why didn't she,}}$ *uh, why didn't he stay at home?*

  and interfere with syntactic parsing
- The Repair is often "roughly" a copy of the Reparandum

  ⇒ *identify repairs by looking for "rough copies"*
- The Reparandum is often short (only 1–2 words long)

  ⇒ word-by-word classifiers can be quite successful
- The Reparandum and Repair can be completely unrelated

# Noisy channel approach to disfluency detection



Source $\Pr(X)$
language model

Source signal $x$
*a flight to Denver on Friday*

Noisy channel $\Pr(U|X)$
introduces disfluencies

Observed noisy signal $u$
*a flight to Boston uh I mean to Denver on Friday*

- Goal: recover the most likely source string $\widehat{x}$ given observed string $u$

$$\widehat{x} \;=\; \underset{x}{\operatorname{argmax}} \Pr(x|u) \;=\; \underset{x}{\operatorname{argmax}} \Pr(u|x)\Pr(x)$$

# The language model

- Given the observed sentence

$$u \quad = \quad \textit{I want a flight to Boston, uh, to Denver on Friday}$$

  the (true) source sentence is

$$x \quad = \quad \textit{I want a flight to Denver on Friday}$$

- The language model estimates $\Pr(x)$
  - here we use a *bigram language model*

$$\begin{aligned}\Pr(x) \quad = \quad & \Pr(I \mid \$) \Pr(\mathit{want} \mid I) \Pr(a \mid \mathit{want}) \Pr(\mathit{flight} \mid a) \\ & \Pr(\mathit{to} \mid \mathit{flight}) \Pr(\mathit{Denver} \mid \mathit{to}) \Pr(\mathit{on} \mid \mathit{Denver}) \\ & \Pr(\mathit{Friday} \mid \mathit{on}) \Pr(\$ \mid \mathit{Friday})\end{aligned}$$

# TAG transducer channel model (1)

- Channel model is a transducer generating *surface:source* pairs $u_i : x_i$

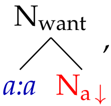
  *a:a flight:flight to:*0 *Boston:*0 *uh:*0 *I:*0 *mean:*0 *to:to Denver:Denver*
- Crossing dependencies ⇒ channel model is a TAG
  - TAG does not reflect grammatical structure (but LM can)
  - right branching finite state model of non-repairs and interregnum
  - adjunction used to describe copy dependencies in repair

# Sample TAG derivation (simplified)

*(I want)* a flight to Boston uh I mean to Denver on Friday …

*Start state:* $N_{want\downarrow}$

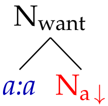
*TAG rule:*



$N_{want}$

*a:a* $N_{a\downarrow}$

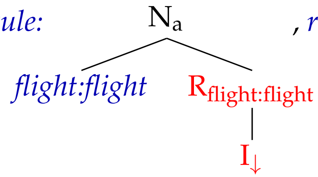
, *resulting structure:*



$N_{want}$

*a:a* $N_{a\downarrow}$

*TAG rule:*

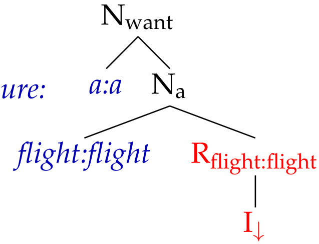


$N_a$

*flight:flight* $R_{flight:flight}$

$I_{\downarrow}$

, *resulting structure:*



$N_{want}$

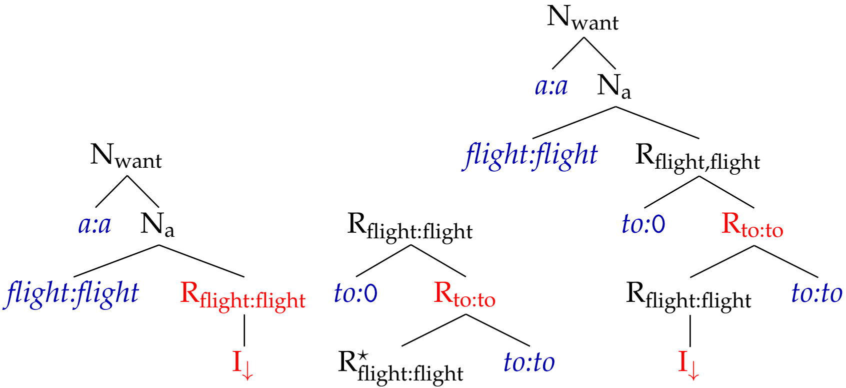
*a:a* $N_a$

*flight:flight* $R_{flight:flight}$

$I_{\downarrow}$

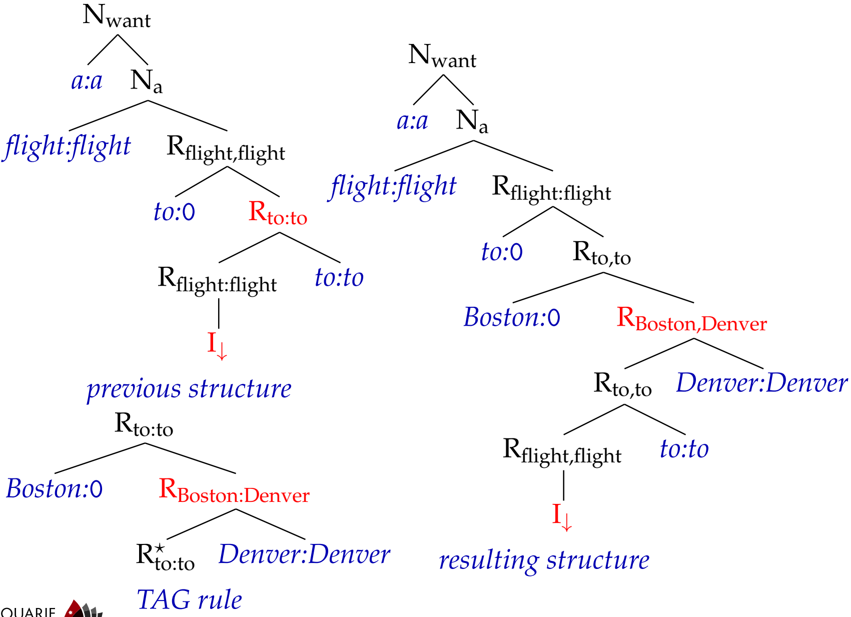
# Sample TAG derivation (cont)

*(I want) a flight* to Boston uh I mean to Denver on Friday …



*previous structure* *TAG rule* *resulting structure*

*(I want) a flight to* Boston uh I mean *to* Denver on Friday …

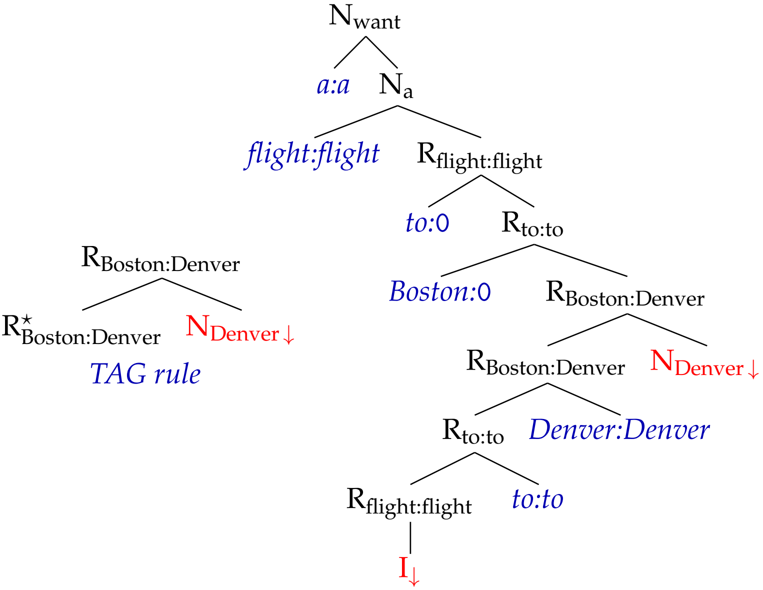


*previous structure*

*TAG rule*

*resulting structure*

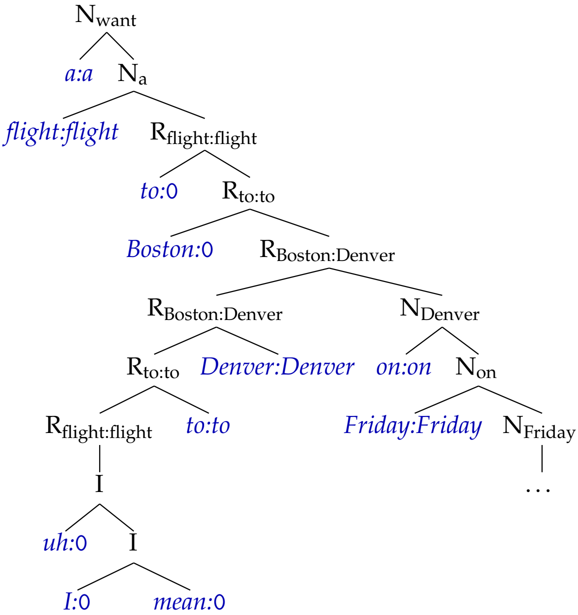
*(I want) a flight to Boston* uh I mean *to Denver* on Friday …


$\mathrm{R}_{\mathrm{Boston:Denver}}$
$\mathrm{R}^{\star}_{\mathrm{Boston:Denver}}$
$\mathrm{N}_{\mathrm{Denver}\downarrow}$
$\mathrm{N}_{\mathrm{want}}$
*a:a*
$\mathrm{N}_{\mathrm{a}}$
*flight:flight*
$\mathrm{R}_{\mathrm{flight:flight}}$
*to:0*
$\mathrm{R}_{\mathrm{to:to}}$
*Boston:0*
$\mathrm{R}_{\mathrm{Boston:Denver}}$
$\mathrm{R}_{\mathrm{Boston:Denver}}$
$\mathrm{N}_{\mathrm{Denver}\downarrow}$
$\mathrm{R}_{\mathrm{to:to}}$
*Denver:Denver*
$\mathrm{R}_{\mathrm{flight:flight}}$
*to:to*
$\mathrm{I}_{\downarrow}$

*TAG rule*

*resulting structure*

*(I want) a flight to Boston* uh I mean *to Denver* on Friday …



# Training Data

- Switchboard corpus (1.3M training words) annotates reparandum, interregnum and repair (we ignore punctuation and partial words)

```
 I/PRP want/VBP a/DT flight/NN [to/TO Boston/NNP ,/, + {F uh/UH ,/, }
      {E I/PRP mean/VBP ,/, } to/TO Denver/NNP] on/IN Friday/NNP
```

  - 5.4% of words are in a reparandum
  - 31K repairs, average length: 1.6 words
- Reparandum and repair word-aligned by minimum-edit-distance, prefers identity, POS identity, similar POS
- Of the 57K alignments in the training data:
  - 35K (62%) are identities
  - 7K (12%) are insertions
  - 9K (16%) are deletions
  - 5.6K (10%) are substitutions (5% with same POS)

# Dynamic programming algorithm for noisy channel

$$\textit{I want a flight } \underbrace{\textit{to Boston,}}_{\text{reparandum}} \underbrace{\textit{uh, I mean}}_{\text{interregnum}} \underbrace{\textit{to Denver}}_{\text{repair}} \textit{ on Friday}$$

- The most likely analysis $\widehat{x}$ generated by the noisy channel model (bigram language model + TAG channel model) can be found using dynamic programming
- Charniak and Johnson (2004) propose a $O(n^5)$ algorithm that involves updating a table with entries of the form

$$\langle \textit{reparandum start}, \textit{reparandum end}, \textit{repair start}, \textit{repair end} \rangle$$

  together with standard bigram trellis entries
- The table entries can be computed in bottom-up left-to-right order

⇒ an incremental version of the Charniak and Johnson model

# Bottom-up restricts incrementality

$$\textit{I want a flight } \underbrace{\textit{to Boston,}}_{\text{reparandum}} \underbrace{\textit{uh, I mean}}_{\text{interregnum}} \underbrace{\textit{to Denver}}_{\text{repair}} \textit{ on Friday}$$

- The model's two basic assumptions:
  1. The repair looks like the reparandum
  2. A sentence without the disfluency is fluent

  don't hold until the disfluency has been completed

  *I want a flight* *to Boston, uh, I mean,* *to …*

  - *to Boston* does not (yet) look very much like *to*
  - taking the disfluency out, there is no fluent continuation (yet)
- Pure bottom-up computation delays until the disfluency has completed and the continuation seen

# Increasing incrementality with speculative completion

- We modify the algorithm to speculatively complete an incomplete repair
    - *incremental completion substitution* assumes that unanalysed words in the reparandum are substitions of (as yet unseen) words in the repair
    - the probability is calculated by summing over all possible repair word substitions
- When the actual following words are observed, we replace the speculatively completed chart cells with their true values

⇒ A disfluency detected by speculative completion may be revised as following words are observed

MACQUARIE
UNIVERSITY

# Evaluating disfluency detection

$$\textit{I want a flight } \underbrace{\textit{to Boston,}}_{\text{reparandum}} \underbrace{\textit{uh, I mean}}_{\text{interregnum}} \underbrace{\textit{to Denver}}_{\text{repair}} \textit{ on Friday}$$

- Fluent words are much more common than disfluent words
  - ⇒ percent correct is not very informative
  - ⇒ prior work reports *f-score* of fluent/disfluent labels (or other metrics)
- At the end of the sentence, the incremental algorithms produce same analyses as Charniak/Johnson algorithm
  - ⇒ Incremental algorithms achieve same f-score (0.778) as Charniak/Johnson algorithm

MACQUARIE UNIVERSITY

# Time to detection evaluation

$$\textit{I want a flight } \underbrace{\textit{to Boston,}}_{\text{reparandum}} \underbrace{\textit{uh, I mean}}_{\text{interregnum}} \underbrace{\textit{to Denver}}_{\text{repair}} \textit{ on Friday}$$

- Time to detection evaluates how quickly an algorithm proposes a disfluency
  - *average time to detection:* average number of words from start of reparandum to when repair is first detected
- Time to detection results:
  No speculation: 5.1 words, with speculation: 4.6 words

  ⇒ *speculation speeds disfluency detection by 0.5 words on average*

MACQUARIE
UNIVERSITY

# Delayed f-score at *k* words

$$\textit{I want a flight } \underbrace{\textit{to Boston,}}_{\text{reparandum}} \underbrace{\textit{uh, I mean}}_{\text{interregnum}} \underbrace{\textit{to Denver}}_{\text{repair}} \textit{ on Friday}$$

- Delayed f-score at *k* words forces the model to label each word as fluent/disfluent when it has seen *k* additional words
  - *delayed f-score at k words:* f-score evaluated when input is *k* words beyond word evaluated
- Delayed f-score results:

| *k* tokens back | **1** | **2** | **3** | **4** | **5** | **6** |
|---|---|---|---|---|---|---|
| No speculation | 0.500 | 0.558 | 0.631 | 0.665 | 0.701 | 0.714 |
| With speculation | 0.578 | 0.633 | 0.697 | 0.725 | 0.758 | 0.770 |

⇒ *Speculation does not decrease accuracy of disfluency detection*

MACQUARIE UNIVERSITY

# Conclusion and future work

- It's possible to develop an incremental version of the Charniak/Johnson disfluency detection algorithm
  - Speculative completion speeds disfluency detection without decreasing accuracy
- Future work:
  - develop a version that does not require sentence-segmented input
  - develop models that detect disfluencies even earlier
  - replace the bigram language model with an incremental parsing model
  - develop methods for training disfluency models from data without disfluency annotations
  - couple this with an incremental speech recogniser

**Interested in statistical models for computational linguistics?**
We're recruiting **PhD students**!.

Contact **Mark.Johnson@mq.edu.au** or **Katherine.Demuth@mq.edu.au**
for more information.

MACQUARIE
UNIVERSITY