BBN+UMD Disfluency Detection

Rich Schwartz
BBN

Matthew Snover, Bonnie Dorr
University of Maryland College Park
Motivation

- To detect disfluencies using simple lexical rules.

- A large number of Fillers ("uh", "um", "you know") and Edits (restarts, repeats) could be identified by examining just the lexemes and part of speech.

- Find a set of rules which describe the annotation of Fillers and Edits. E.g.,
  - The word "uh" is usually a Filler
  - Words that are repeated are usually Edits
Outline

- Procedure
- Results
- Error Analysis
- Discussion
Transformation Based Learning

- Automatically induce rules from the training data.
- TBL is rule induction system (Brill 1995)
- Start with initial hypothesis
  - Disfluency detection: All words are fluent
- Greedily learn set of rules that modify hypothesis to reduce the error rate
  - All filled pauses are fillers
  - Left side of repeat is edit
  - If “I” is followed by “You” then “I” is an edit
- Possible rules are generated by expanding rule templates.
  - All $X \overline{L}_1$ (e.g., All “UH” are Filler; All “you know” are Fillers)
  - Left side of repeat $\overline{L}_1$ (e.g. Edit)
  - If $X$ is followed by $Y$, $\overline{L}_1$
- Output is set of rules, which can then be applied to test data.
Feature Set

- Lexeme (The word itself)
- Part Of Speech (Max Entropy Tagger)
- Silence following word (according to time alignment)
- High Frequency Word for Speaker
  - e.g.: Speaker uses word “like” a lot

- 3 Target “Tags”: FLUENT, EDIT, FILLER
Frequent Word Detection

- “Like” is only a disfluency 22% of time.
- If speaker uses “like” much more often than is common, then “like” is probably not being used in a fluent way.
- We find speakers who use “like” very often, and the system finds rules that mark “like”’s for that speaker as disfluent.
- Also works for other less common disfluencies such as:
  - “so” (disfluent 30%)
  - “actually” (disfluent 45%)
### Sample Templates and Rules

<table>
<thead>
<tr>
<th>Rule Template</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>(_{l_1} w_x, _{l_2} w_x)\</td>
<td>([<em>{\text{Fluent} w</em>{FP}}, <em>{\text{Filler} w</em>{FP}}]\</td>
</tr>
<tr>
<td>([uh_{FP}]\</td>
<td></td>
</tr>
<tr>
<td>([X\ Y] _{l_1} X\ Y)\</td>
<td>([\text{you know}] _{\text{Filler} you know}\</td>
</tr>
<tr>
<td>([\text{you know}]\</td>
<td></td>
</tr>
<tr>
<td>(Z \ X Y\ _{l_1} X Y)\</td>
<td>\text{do } [\text{you know}] _{\text{Fluent you know}}\</td>
</tr>
<tr>
<td>(\text{do } [\text{you know}]\</td>
<td></td>
</tr>
<tr>
<td>(_{l_1} w_x) &lt;p&gt; w_Y _{l_2} w_x) &lt;p&gt; w_Y\</td>
<td>([<em>{\text{Fluent} w</em>{DT}}] &lt;p&gt; w_{DT} _{\text{Edit} w_{DT}}] &lt;p&gt; w_{DT}\</td>
</tr>
<tr>
<td>([\text{the_{DT}}] &lt;p&gt; a_{DT}\</td>
<td></td>
</tr>
<tr>
<td>(_{l_1} w_Y\ _{l_2} w_Y\ w_z\</td>
<td>(w_{&lt;s&gt;} [<em>{\text{Fluent} w</em>{PRP}}] w_{PRP} _{\text{Edit} w_{PRP}}] w_{PRP}\</td>
</tr>
<tr>
<td>(&lt;s&gt;<em>{&lt;s&gt;} [he</em>{PRP}] she_{PRP}\</td>
<td></td>
</tr>
<tr>
<td>([_{A}\ w_x B^* A^* _{l_1} A^<em>] w_x B^</em> A^*\</td>
<td>([<em>{A^<em>} w_x B^</em> A^* _{\text{Edit} A^*} w</em>{FP} B^* A^*]\</td>
</tr>
<tr>
<td>([\text{car}] uh_{FP} \text{ red car}\</td>
<td></td>
</tr>
</tbody>
</table>
Training

- Used RT03F 1st, 2nd, 3rd thirds of training data
- BNews ~190k tokens
- CTS ~ 490k tokens
- Separately trained BNews and CTS systems

- All training was on reference transcripts.
- No training on STT output.

- Both systems had 33 rule templates (>10^{13} possibilities)
- On BNews TBL learned 19 rules (56,000 > Threshold)
- On CTS TBL learned 106 rules (99,000 > Threshold)
Top Rules (CTS)

1. All Filled Pauses: Fluent → Filler
2. Left Side of Repeat is Edit
3. You Know: Both are Fillers
4. Well with ‘UH’ POS is Filler
5. All Fragments are Edits
6. I Mean: Both are Fillers
7. Left Side of Repeat Separated by FP is Edit
8. Left Side of Repeat Separated by Fragment is Edit
9. All Filled Pauses: Edits → Fillers
10. Fragments at end of sentence: Edit → Fluent
11. A* PRP B* A*: First A* is Edit
12. PRP followed by PRPVB: Fluent → Edit
IP Detection

- IP detection is completely dependent upon disfluency annotation

- IPs were assigned according to these simple rules:
  1. IP assigned before each sequence of fillers
  2. IP assigned before each filled pause filler
  3. IP assigned after each sequence of edits
## RT-Eval Results

<table>
<thead>
<tr>
<th></th>
<th>Edits</th>
<th>Fillers</th>
<th>IPs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CTS Reference</strong></td>
<td>68.0%</td>
<td>18.1%</td>
<td>41.1%</td>
</tr>
<tr>
<td><strong>CTS STT</strong></td>
<td>87.9%</td>
<td>48.8%</td>
<td>69.0%</td>
</tr>
<tr>
<td><strong>BNews Reference</strong></td>
<td>45.3%</td>
<td>6.5%</td>
<td>18.5%</td>
</tr>
<tr>
<td><strong>BNews STT</strong></td>
<td>94.5%</td>
<td>78.8%</td>
<td>86.7%</td>
</tr>
<tr>
<td></td>
<td>93.9%*</td>
<td>57.2%*</td>
<td>70.1%*</td>
</tr>
</tbody>
</table>
In development BNews data, recognizer was generating too many “UH”s, so these were stripped out as a post process. The system did not over generate for evaluation data, so we missed all the “UH”s.

Not stripping out the UHs gives 57.2% filler error (versus 78.8% submitted).
Speech vs. Reference

• Why the large difference between speech and reference error rates?

• System trained only on reference data.
  – Did not learn to correct for recognizer error.

• Percentage of errors when wrong words were output
  – CTS Edit 27% of error (87.9% → 64.1%)
  – CTS Filler 19% of error (48.8% → 39.53%)

• Percentage of errors when no words were output
  – CTS Edit 19% of error (87.9% → 71.2%)
  – CTS Filler 12% of error (48.8% → 42.9%)
Errors

• System misses long edits.
  – [And whenever they come out with a warning (n-)] you know they were (c-) coming out with a warning about (trains).
  – [Most of the people most of my aunts and uncles and everything have] (we’ve) never really had ...
  – Difficult to detect since edit itself appears fluent.
  – Accounts for 48% of edit errors (CTS Reference)

• The system is good at detecting regular localized disfluencies, but has problems with longer dependencies.
• The system is also sensitive to errors in SU detection.
Discussion

- Transformation Based Learning approach to annotating disfluencies using primarily lexemes and part of speech.

- Speaker dependent word frequency useful for distinguishing rarer disfluent words.

- Reference annotation is very different from recognizer output.

- Error counting for STT output penalizes the system for many recognizer errors.
  - Failure to recognize an edit word can cause entire edit to not be detected.
  - If a filled pause is hallucinated, it will be labelled as a filler and ‘removed’. The result is the same as if we didn’t hallucinate the filled pause, but we are scored incorrect.
Future Work

• Improve disfluency detection
  – Consider more global error measures, such as parsing
    • Parser (trained on fluent speech) might parse disfluencies poorly.
    – Include LM information as a feature.

• Use disfluency modeling to ‘clean up’ transcript.

• Primary Focus: Use model of disfluencies to reduce WER