Automatic Disfluency Detection in Multi-party Conversations

Feast, 30th September 2009
Sebastian Germesin
Outline

• Motivation
• Theoretical Background
• Data (AMI Corpus)
• Disfluency Detection System
  • Hybrid Classification Approach
  • Self-arranging Modules
  • Experimental Results
• Conclusions & Outlook
Motivation

Example

SEE, I THINK WE SHOULD GIVE THE REMOTE CONTROL A MASCOT, SOMETHING LIKE AN A SNOW LEOPARD.

YEAH, BUT THEN, WE UH SHOULD PAINT IT IN IN YOU KNOW WHITE COLOUR.
Motivation

• Have to detect (and clean) disfluencies in the transcribed speech
  • Readability
    • Transcription
    • Extractive Summarization
  • Post-Processing
    • NLP-systems’ performance drop when faced with disfluent speech

• Human detector?
  • Too expensive!
  • Too slow!

⇒ Automatic Detection System!
“Disfluencies are syntactical and grammatical [speech] errors that occur in spoken but not in written language.” [Besser, 2006]
“The cat uh the dog sneaks around the corner.”
“The cat uh the dog sneaks around the corner.”
Theoretical Background

Terminology

Reparandum

"The cat uh the dog sneaks around the corner."
Theoretical Background

Terminology

Reparandum

Reparans

Interregnum

complex

“The cat uh the dog sneaks around the corner.”
Theoretical Background

Terminology

“\textit{The} d \textit{dog} sneaks around the corner.”

Repandandum
Theoretical Background

All Types

- Disfluency
  - Uncorrected
    - Mistake
    - Omission
    - Order
  - Revision
    - Deletion
    - Insertion
    - Repetition
    - Replacement
  - Other
- Deleteable
  - Hesitation
  - Stuttering
  - Slip Of the Tongue
  - Discourse Marker
  - Disruption
- Simple disfluencies
Data

quantitative

• AMI meeting corpus
  • 135 meetings (~ 100 hours speech)
  • 4 participants
  • task: design a remote control
  • freely interaction
  • Many annotations, e.g.:
    • Transcribed speech
    • Dialogue acts
    • Gestures
    • ...
Data

quantitative

- 45 meeting enriched with disfluency annotation
  - 31,000 Disfluencies
  - 15.8% erroneous words
  - 41.5% disfluent Dialogue Acts
  - 80% (33) for training
  - 20% (12) for evaluation
Data qualitative

• Discovered a heterogeneity towards the strictness of different disfluency types
  1. Some disfluencies have strict structure
     • ex.: Repetition : “The cat the cat plays “
  2. Some other disfluencies have also strict structure but this structure is very common in natural language
     • ex.: Replacement : “The dog the cat plays“
     • ex.: Fluent : “The dog the cat and the bird play”
  3. Some other disfluencies have no obvious structure
     • ex.: Disruptions : “The dog the cat and“
     • ex.: Order : “The plays cat”
Automatic System

Design Question

• Can we leverage the heterogeneity of disfluencies for their detection?
  → Yes!
    → Use modules for subsets of disfluencies
    → Use different feature-sets for each module (depending on the disfluency types)
    → Find “optimal” classifier for each module
Automatic System

Hybrid Modules

- **SHS:**
  - Stuttering, Hesitation, Slip-of-the-Tongue
- **REP:**
  - Repetition
- **DNE:**
  - Discourse Marker, Explicit Editing Term
- **DEL:**
  - Deletion
- **REV:**
  - Insertion, Replacement, Restart, Other
How to combine the modules?

Classifier Library
- Rules
- MaxEnt
- J48
- ...
- ...

Modules
- SHS
- REP
- DNE
- DEL
- REV

Corpus

Training

trained System
- $\text{SHS}_{J48}$
- $\text{REP}_{\text{Rule}}$
- $\text{RE}_{\text{Rule}}$
- $\text{DNE}_{\text{Max}}$
- $\text{DNE}_{J48}$
- $\text{REP}_{J48}$
Training Process

Self-arranging Modules

• Immense search space
  • \(#(\text{modules}) \times #(\text{classifier}) \times \text{placeInSystem}\)

• Solution(s):
  • **Old** system:
    • Chosen manually
  • **Current** system:
    • Automatically trained
  1. Use greedy hill-climbing
    – Use weight for errors to improve Precision!
Algorithm 1 Greedy hill-climbing process of self-arranging modules

system ← empty List

repeat
    bestPerformance ← 0.0
    bestModule ← null
    for all Module m : modules do
        for all Classifier c : classifier do
            train(m_c)
            evaluate(m_c)
            performance_a(m_c) ← correct(m_c) - α * errors(m_c)
            if performance_a(m_c) > bestPerformance then
                bestPerformance ← goodness_a(m_c)
                bestModule ← (m_c)
            end if
        end for
    end for
    if bestModule ≠ null then
        system.add(bestModule)
    end if
until bestModule = null
return system
Training Process

Self-arranging Modules

- Immense search space
  - $(\text{#(modules)} \times \text{#(classifier)} \times \text{placeInSystem})$

- Solution(s):
  - **Old** system:
    - Chosen manually
  - **Current** system:
    - Automatically trained
      1. Use greedy hill-climbing
        - Use weight for errors to improve Precision!
      2. Reduce classifier library
        - Take 10% results in maximal performance loss of 2.3% (depending on the module)
GroDi

Performance-Curve of J48

Best: J48 "-L -U -M 2 -A"
## Experimental Results

<table>
<thead>
<tr>
<th>System</th>
<th>Train. data</th>
<th>Eval. data</th>
<th>Accuracy</th>
<th>avg. F1</th>
<th>RT-factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline</td>
<td>--</td>
<td>6 m.</td>
<td>90.3 %</td>
<td>85.7 %</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12 m.</td>
<td>88.6 %</td>
<td>83.3 %</td>
<td></td>
</tr>
<tr>
<td>old</td>
<td>22 m.</td>
<td>6 m.</td>
<td>92.9 %</td>
<td>90.5 %</td>
<td>0.42</td>
</tr>
<tr>
<td>new</td>
<td>22 m.</td>
<td>6 m.</td>
<td>95.3 %</td>
<td>94.8 %</td>
<td></td>
</tr>
<tr>
<td></td>
<td>33 m.</td>
<td>6 m.</td>
<td>95.1 %</td>
<td>94.7 %</td>
<td>0.11</td>
</tr>
<tr>
<td></td>
<td></td>
<td>12 m.</td>
<td>94.5 %</td>
<td>93.5 %</td>
<td></td>
</tr>
</tbody>
</table>
Conclusions

• Aims:
  • Development of a system that automatically detects a broad set of disfluencies
  • Fully automatic learning process
  • Robust and Fast

• Achievements:
  • Stand-alone tool for detection of disfluencies: GroDi - Get rid of Disfluencies
  • Self-arranging modules
  • Detection rate: 95% Accuracy
  • Real-time factor of 0.11
Outlook

• Develop module(s) for the detection of *Mistake, Order, Omission*

• Embed other learning approaches, e.g.:
  • Conditional Random Fields
  • HMMs

• Use other corpus like, e.g., Switchboard
Thank you!
Demo?

---

UI: then they're small cute and furry, and uh when planets of the apes becomes real, I'm gonna be up there with them.

ME: I'm getting kind of restlessness, can you sit down, we're starting to talk, I got some further information from their family, and yeah that they have lots of personality and uh be fit, and in robust.

ID: that's too much gear.

---

Segment Classification + Disfluency Detection:

UI: no, yeah.

ME: inf: you just got the project management about what the project is designing a remote control that's about 1, didn't get anything did you get the same thing.

ME: off: well that's fine.

ME: inf: is this one here right.

ME: inf: is that what everybody got okay, so we've gone have like individual work, and then a meeting about it, and expect that process three time are set, and at this point we get to put the whiteboard over there.

ME: ass: okay, now, alright, my favorite animal.

ME: elinf: is like.

ME: inf: a beagle.

ME: ass: yeah, yeah.

ME: ass: light, lovely.

UI: inf: well my favorite animal would be a monkey.

ID: ass: cool.

Meeting progress: 10h 04m 50s.
# GroDi

## Diff. Module Arrangements

<table>
<thead>
<tr>
<th>Nr.</th>
<th>Module</th>
<th>Classifier</th>
<th>AMI</th>
<th>Module</th>
<th>Classifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>SHS</td>
<td>J48 ’-U -M 2’</td>
<td>SHS</td>
<td>J48 ’-L -U -M 3 -A’</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>REP</td>
<td>RuleMatcher ”</td>
<td>REP</td>
<td>RuleMatcher ”</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>DNE</td>
<td>MaxEntStanford ”</td>
<td>DNE</td>
<td>MaxEntStanford ”</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>REP</td>
<td>RuleMatcher ”</td>
<td>REP</td>
<td>RuleMatcher ”</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>DNE</td>
<td>J48 ’-L -S -C 0.45 -M 2 -A’</td>
<td>DNE</td>
<td>J48 ’-U -M 5’</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>SHS</td>
<td>J48 ’-U -B -M 4’</td>
<td>SHS</td>
<td>J48 ’-L -U -M 4’</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>REP</td>
<td>RuleMatcher ”</td>
<td>REP</td>
<td>RuleMatcher ”</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>DNE</td>
<td>J48 ’-U -M 3’</td>
<td>DEL</td>
<td>RuleMatcher ”</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>DNE</td>
<td>MaxEntStanford ”</td>
<td>DNE</td>
<td>J48 ’-S -R -N 9 -Q 1 -M 2’</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>DEL</td>
<td>RuleMatcher ”</td>
<td>DNE</td>
<td>J48 ’-L -U -B -M 3’</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>SHS</td>
<td>J48 ’-U -M 2 -A’</td>
<td>REP</td>
<td>RuleMatcher ”</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>DNE</td>
<td>J48 ’-U -M 2’</td>
<td>SHS</td>
<td>J48 ’-L -U -B -M 2 -A’</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>REP</td>
<td>RuleMatcher ”</td>
<td>DNE</td>
<td>J48 ’-U -B -M 2’</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>DNE</td>
<td>J48 ’-L -U -M 3’</td>
<td>REP</td>
<td>RuleMatcher ”</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>SHS</td>
<td>MaxEntStanford ”</td>
<td>DNE</td>
<td>MaxEntStanford ”</td>
<td></td>
</tr>
</tbody>
</table>
GroDi

- Used technologies
  - WEKA toolkit for machine learning
  - Maximum Entropy classifier from Stanford NLP group
  - CRF Tagger from http://crftagger.sourceforge.net/
- Features for machine learning:
  - Lexical: words, lexical parallelism, (POS-Tags)
  - Prosodic: duration, pauses, pitch, energy
  - Dynamic: disfluency types of surrounding words
  - Speaker: age, role in meeting, native language