Presentation for ENLG 2014 paper on Text Simplification using Typed Dependencies

Advaith Siddharthan

Department of Computing Science
University of Aberdeen
Outline

1. Motivation
2. Reformulation using Transfer Rules
   - Gen-light
   - Gen-heavy
   - N-best parses
3. Results and Examples
4. Summary
Motivation for Sentence Reformulation

Text Simplification

- Making the same semantic/pragmatic content [Dorr et al.2004] more accessible through reformulation
  - Reduce lexical or grammatical complexity by replacing difficult words or splitting long sentences
  - Make content more transparent by making discourse relations explicit.
[Siddharthan2010]:

The original police inquiry, which led to Mulcaire being jailed in 2007, also discovered evidence that he has successfully intercepted voicemail messages belonging to Rebekah Brooks, who was editor of the Sun when Mulcaire was working exclusively for its Sunday stablemate.

The original police inquiry led to Mulcaire being jailed in 2007. The police inquiry also discovered evidence that he has successfully intercepted voicemail messages belonging to Rebekah Brooks. Rebekah Brooks was editor of the Sun. This was when Mulcaire was working exclusively for its Sunday stablemate.
Research Objective

- Improve robustness of syntactic simplification to parsing error.
- Approaches [Siddharthan2011]:
  - Using multiple (n-best) parses
  - Using a surface realiser
Research Objective

- Improve robustness of syntactic simplification to parsing error.
- Results:
  - Using multiple (n-best) parses helps
  - Using a surface realiser introduces more infelicities
**Uses of Automatic Simplification**

- **Literacy Aids**
  - Simplify online texts to linguistic levels suitable for adult second-language learners.
    - Univ. of Washington (Petersen and Ostendorf, 2007)
    - 87% good choice of split point (Siddharthan 2003 system)
    - $R = .37$, compared to sentences in manual simplification

- **Enabling Access to Information**
  - Basis for a project on text simplification in Portuguese
    - Univ. of São Paulo (Aluísio et al., 2008, etc.)
    - Aims to make online text more accessible to Brazilians with low literacy skills
    - Considers adults with 4-8 years of schooling
**Uses of Text Simplification**

- **Education and Literacy**
  - Literacy skills improve faster when
    - Reading material is interesting
    - Language is challenging without being intimidating
  - Simplified texts commonly used in language learning
  - Manual simplification helps
    - Low ability readers outperform middle ability readers on comprehension tests (L’Allier, 1980; Linderholm et al., 2000)
  - Automatic simplification not quite there yet...

- **Enable access to information for disadvantaged groups**
  - Language deficits (from aphasia, deafness, aging, etc.)
This research

- Transformation rules applied to Stanford Dependencies
- Two approaches to generation:
  - Gen-light: As in Siddharthan (2010): Reuse original word order and morphology; specify changes in transfer rules
  - Gen-heavy: Transform Stanford Deps to DSyntS and generate using RealPro.
- Two parsing modes
  - Single Parse
  - N-best Parse
Generation from Typed Dependencies

- **Phrasal Parse Trees**
  - Generation is depth first l-r search for nodes

- **Dependency Trees**
  - Generation is inorder: left subtrees, root, right subtrees
  - Order to process involves typical generation decisions
  - guided by type and statistical preferences for word/phrase order
  - But we can use word order from original sentence

The/DT cat/NN was/VBD chased/VBN by/IN the/DT dog/NN ./

det(cat-2, The-1)
nsubjpass(chased-4, cat-2)
auxpass(chased-4, was-3)
det(dog-7, the-6)
agent(chased-4, dog-7)
punct(chased-4, .-8)
Generation from Typed Dependencies

- **Phrasal Parse Trees**
  - Generation is depth first l-r search for nodes

- **Dependency Trees**
  - Generation is inorder: left subtrees, root, right subtrees
  - Order to process involves typical generation decisions
  - guided by type and statistical preferences for word/phrase order
  - But we can use word order from original sentence

The/DT cat/NN was/VBD chased/VBN by/IN the/DT dog/NN ./.

det(cat-2, The-1)
nsubjpass(chased-4, cat-2)
auxpass(chased-4, was-3)
det(dog-7, the-6)
agent(chased-4, dog-7)
punct(chased-4, .-8)

chased:4
nsbjpass
auxpass
agent
cat:2 was:3 dog:7
det
det

The:1 the:6
Transfer Rules: Gen-light

1. Match and Delete:
   1. nsubjpass(??X0, ??X1)
   2. auxpass(??X0, ??X2)
   3. agent(??X0, ??X3)

2. Insert:
   1. nsubj(??X0, ??X3)
   2. dobj(??X0, ??X1)

\[
\text{cat:} 2 \quad \text{was:} 3 \quad \text{dog:} 7 \quad \text{cat:} 2 \quad \text{dog:} 7
\]
\[
\text{The:} 1 \quad \text{the:} 6 \quad \text{The:} 1 \quad \text{the:} 6
\]
Transfer Rules: Gen-light

1. Match and Delete:
   1. nsubjpass(??X0, ??X1)
   2. auxpass(??X0, ??X2)
   3. agent(??X0, ??X3)

2. Insert:
   1. nsubj(??X0, ??X3)
   2. dobj(??X0, ??X1)

3. Traversal Order Specifications:
   1. Node ??X0: [??X3, ??X0, ??X1] chased:4
dobj nsubj
cat:2
dog:7
det
det
The:1 the:6
Transfer Rules: Gen-light

1. Match and Delete:
   1. nsubjpass(??X0, ??X1)
   2. auxpass(??X0, ??X2)
   3. agent(??X0, ??X3)

2. Insert:
   1. nsubj(??X0, ??X3)
   2. dobj(??X0, ??X1)

3. Traversal Order Specifications:
   1. Node ??X0: [??X3, ??X0, ??X1]

4. Node Transformations
   1. Lexical Changes (Agreement, etc)
   2. Node / Subtree Deletions
Transformation rules need 5 lists:

1. CONTEXT: Transform only proceeds if this list of GRs can be unified with the input GRs.
2. DELETE: List of GRs to delete from input.
3. INSERT: List of GRs to insert into input.
4. ORDERING: List of nodes with subtree order specified.
5. NODE-OPERATIONS: List of lexical substitutions and deletion operations on nodes.

Last 2 should really be dealt with by generator?
Outline

1. Motivation

2. Reformulation using Transfer Rules
   - Gen-light
   - Gen-heavy
   - N-best parses

3. Results and Examples

4. Summary
Gen-heavy Approach

1 Conversion from Stanford Deps to DSyntS and generation by RealPro

2 Issues:
   1 In DSyntS, words are presented as lemmas, with tense, voice, aspect, mood, taxis, number, person, gender, etc. represented as features. This means we need to analyse part-of-speech tags, auxiliary verbs and pronouns to provide RealPro with the correct input.
   2 Unlike the Stanford Dependencies that contains 52 fine-grained types, DSyntS uses only the following seven types: ‘I’, ‘II’, ‘III’, ‘IV’, ‘ATTR’, ‘DESC-ATTR’ and ‘APPEND’. Thus, we need to map each of the Stanford dependencies to one of these types. There are some subtleties regarding coordination and relative clauses, but the mapping is for the most part straightforward.
The dog chased the cat.

DSYNTS:
"chase" [class:"verb" voice:"act"
    tense:"past" aspect:"simple"
    taxis:"nil" polarity:"nil"]

  (I "dog" [class:"common_noun"
            number:"sg" article:"def"]
   II "cat" [class:"common_noun"
             number:"sg" article:"def"]
  )

END:
Text Simplification

- Applying multiple transformation rules
  - rules can be applied iteratively
  - order is not important (though there are special cases)

The cat was chased by a dog that was barking

\[\text{det(cat-2, The-1)}\]
\[\text{nsubjpass(chased-4, cat-2)}\]
\[\text{auxpass(chased-4, was-3)}\]
\[\text{det(dog-7, a-6)}\]
\[\text{agent(chased-4, dog-7)}\]
\[\text{nsubj(barking-10, dog-7)}\]
\[\text{aux(barking-10, was-9)}\]
\[\text{rcmod(dog-7, barking-10)}\]

- Apply rule for relative clauses:
  1. Match and Delete:
     2. Insert:

     (a) \text{rcmod(??X0, ??X1)}
     (a) \text{nsubj(??X1, ??X0)}
     (b) \text{nsubj(??X1, ??X0)}
Applying multiple transformation rules

- rules can be applied iteratively
- order is not important (though there are special cases)

The cat was chased by a dog that was barking

\[
\begin{align*}
det(\text{cat-2}, \text{The-1}) \\
nsubjpass(\text{chased-4}, \text{cat-2}) \\
auxpass(\text{chased-4}, \text{was-3}) \\
det(\text{dog-7}, \text{a-6}) \\
agent(\text{chased-4}, \text{dog-7}) \\
nsubj(\text{barking-10}, \text{dog-7}) \\
aux(\text{barking-10}, \text{was-9})
\end{align*}
\]

Apply rule for voice change:

1. Match and Delete:
   (a) nsubjpass(??X0, ??X1) 
   (b) auxpass(??X0, ??X2) 
   (c) agent(??X0, ??X3)
2. Insert:
   (a) nsubj(??X0, ??X3) 
   (b) dobj(??X0, ??X1)
Text Simplification

- Applying multiple transformation rules
  - rules can be applied iteratively
  - order is not important (though there are special cases)

The cat was chased by a dog that was barking

\[\text{det(cat-2, The-1)}\]
\[\text{dobj(chased-4, cat-2)}\]
\[\text{det(dog-7, a-6)}\]
\[\text{nsubj(chased-4, dog-7)}\]
\[\text{aux(barking-10, was-9)}\]
\[\text{nsubj(barking-10, dog-7)}\]
Text Simplification

- Applying multiple transformation rules
  - rules can be applied iteratively
  - order is not important (though there are special cases)

The cat was chased by a dog that was barking

\[
\begin{align*}
\text{det(cat-2, The-1)} & \quad \text{chased} \\
\text{dobj(chased-4, cat-2)} & \quad \text{dobj} \\
\text{det(dog-7, a-6)} & \quad \text{nsubj(chased-4, dog-7)} \\
\text{nsubj(chased-4, dog-7)} & \quad \text{aux(barking-10, was-9)} \\
\text{aux(barking-10, was-9)} & \quad \text{nsubj(barking-10, dog-7)} \\
\text{nsubj(barking-10, dog-7)} & \quad \text{det} \\
\end{align*}
\]

A dog chased the cat. The dog was barking.
Outline

1 Motivation

2 Reformulation using Transfer Rules
   - Gen-light
   - Gen-heavy
   - N-best parses

3 Results and Examples

4 Summary
Most errors in output could be traced back to inaccurate parsing

- Cars and pick-up trucks with badly twisted and still smouldering frames littered the three compounds, which housed villas and four-storey blocks.
- The three compounds housed. Cars and pick-up trucks with badly twisted and still smouldering frames littered the compounds, villas and four-storey. And Cars blocks.

We use the n-best parses and try to rank the simplified texts
Evaluating output quality

- Manual identification of patterns of infelicities:
  - Sentences ending in subject pronouns, prepositions or conjunctions
  - Word repetition (e.g., “is is” or “to to”)
  - Prepositions followed by subject pronouns (e.g., “of he”)
  - Bad sequences of conjunctions and prepositions (e.g., “because but” or “until for”)

- Positive weight for overlap with original sentence
  - Bigram and trigram overlap with original sentence (as a fraction)
  - The number of sentences in the output (this is to encourage the application of simplification rules)
  - A bonus if the simplification was performed on the top-ranked parse (as this is the most likely to be correct)
Evaluation Criterion

- **Extent:** The level of simplification achieved, based on the number of transforms performed and the average sentence length in the simplified text.

- **Precision of rules:** The proportion of transformed sentences for which the rules have been applied accurately (as judged by developer), so that the output is grammatical with (a) correct verb agreement and inflexion and (b) modifiers/complements appearing in acceptable orders.

- **Acceptability of output:** Here we test the quality of output using evaluators who do not have knowledge of the transformation rules.
### Results

<table>
<thead>
<tr>
<th>System Setting</th>
<th>Av S Len</th>
<th>#Trans/S</th>
<th>%S Trans</th>
<th>%Acc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td>20.9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>gen-l/1 parse</td>
<td>15.3</td>
<td>0.65</td>
<td>50.2</td>
<td>83.9</td>
</tr>
<tr>
<td>gen-l/50 parses</td>
<td>14.3</td>
<td>0.74</td>
<td>55.4</td>
<td>87.9</td>
</tr>
<tr>
<td>gen-h/1 parse</td>
<td>14.8</td>
<td>0.65</td>
<td>50.2</td>
<td>70.8</td>
</tr>
<tr>
<td>gen-h/50 parses</td>
<td>14.0</td>
<td>0.74</td>
<td>55.4</td>
<td>77.7</td>
</tr>
</tbody>
</table>

**Table**: Test results for four configurations of the system: gen-light and gen-heavy in single parse and 50-best parses modes. The columns report average sentence length in words, average number of transformations performed on each input sentence, percentage of input sentences with at least one transformation, the correctness of the transformations.
Results

<table>
<thead>
<tr>
<th>Judge 1</th>
<th>Judge 2</th>
<th>$\kappa$</th>
<th>% Agreement</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>.55</td>
<td>78%</td>
</tr>
<tr>
<td>A</td>
<td>Developer</td>
<td>.52</td>
<td>79%</td>
</tr>
<tr>
<td>B</td>
<td>Developer</td>
<td>.32</td>
<td>68%</td>
</tr>
</tbody>
</table>

Table: Pairwise agreement on acceptability

<table>
<thead>
<tr>
<th>System</th>
<th>% Acceptable</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>B</td>
</tr>
<tr>
<td>gen-l/1 parse</td>
<td>.59</td>
</tr>
<tr>
<td>gen-l/50 parses</td>
<td>.62</td>
</tr>
<tr>
<td>gen-h/1 parse</td>
<td>.19</td>
</tr>
<tr>
<td>gen-h/50 parses</td>
<td>.20</td>
</tr>
</tbody>
</table>

Table: The percentage of transformed sentences acceptable to the three raters (the developer and two judges) for 4 reformulations each of 50 sentences. The final column treats a transformation as acceptable if at least 2 raters find it acceptable.
Examples

Accurate and Inaccurate Transformations

1. I am very relieved to have won my appeal and for recognition I was treated unfairly and unlawfully.
   √ I am very relieved to have won my appeal. And for recognition I was unfairly and unlawfully treated.
   × I am very relieved to have won my appeal. And I was unfairly and unlawfully treated for recognition.

2. It is believed to include recordings Mulcaire made of messages left on Rice’s mobile phone, including several from friends and families.
   √ It is believed to include recordings Mulcaire made of messages left on Rice’s mobile phone. This includes several from friends and families.
   × It is believed to include recordings Mulcaire, made of messages, left on Rice’s mobile phone. This includes several from friends and families.

Table: Examples of automatic reformulations.
Conclusions

- Robustness improved by using n-best parses in overgenerate-and-rank approach.
- Using existing generator (RealPro) is brittle:
  - Misanalyses by the parser can result in unacceptable word and constituent orders in the generated texts.
  - This problem would, we believe, be overcome if the generator could make use of word and phrase order from the input sentence.
- In future:
  - Combine with summarisation task to generate abridged texts.
  - Evaluate with groups known to have reading difficulties.
  - Look at Lexical Simplification (see poster).
Bonnie Dorr, David Farwell, Rebecca Green, Nizar Habash, Stephen Helmreich, Eduard Hovy, Lori Levin, Keith Miller, Teruko Mitamura, Owen Rambow, et al.
2004.
Semantic Annotation and lexico-syntactic paraphrase. LREC.

Advaith Siddharthan.
2010.
Complex lexico-syntactic reformulation of sentences using typed dependency representations.

Advaith Siddharthan.
2011.
Text simplification using typed dependencies: A comparison of the robustness of different generation strategies.
In Proceedings of the 13th European Workshop on Natural Language Generation (ENLG 2011), Nancy, France.