Text Simplification using Typed Dependencies: A Comparison of the Robustness of Different Generation Strategies

Advaith Siddharthan
University of Aberdeen
Department of Computing Science
advaith@abdn.ac.uk

Abstract

We present a framework for text simplification based on applying transformation rules to a typed dependency representation produced by the Stanford parser. We test two approaches to regeneration from typed dependencies: (a) **gen-light**, where the transformed dependency graphs are linearised using the word order and morphology of the original sentence, with any changes coded into the transformation rules, and (b) **gen-heavy**, where the Stanford dependencies are reduced to a DSyntS representation and sentences are generating formally using the RealPro surface realiser. The main contribution of this paper is to compare the robustness of these approaches in the presence of parsing errors, using both a single parse and an n-best parse setting in an overgenerate and rank approach. We find that the gen-light approach is robust to parser error, particularly in the n-best parse setting. On the other hand, parsing errors cause the realiser in the gen-heavy approach to order words and phrases in ways that are disliked by our evaluators.

1 Introduction

In this paper, we present a system, REGENT, for text regeneration tasks such as text simplification, style modification or paraphrase. Our system applies transformation rules specified in XML files, to a typed dependency representation obtained from the Stanford Parser (De Marneffe et al., 2006). There are currently rule files for simplifying coordination (of verb phrases and full clauses), subordination, apposition and relative clauses, as well as conversion of passive to active voice; for instance, simplifying:

> 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.

to:

> 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.

The main aim of this paper is to describe and compare two methods for generating sentences from the transformed dependency graphs:

1. **gen-heavy:** We use RealPro (Lavoie and Rambow, 1997), a statistical realiser to generate, making all decisions related to morphology and word ordering.
2. **gen-light:** We reuse word order and morphology from the original sentence, and specify any changes to these as part of each transformation rule.

Both options have pros and cons. In the gen-light approach described in detail in Siddharthan (2010) and summarised in §3.1, we can reuse information from the input sentence as much as possible, leading to very efficient generation. The downside is

Proceedings of the 13th European Workshop on Natural Language Generation (ENLG), pages 2–11,
Nancy, France, September 2011.

that we need to encode some generation decisions within transfer rules, making them cumbersome to write and difficult to learn automatically. A case can be made, particularly for the issue of subject-verb agreement, for such issues to be handled by a generator. This would make the transfer rules simpler to write, and indeed easier to learn automatically in a supervised setting. While many syntactic simplification rules are quite easy to formulate by hand, this might be an important consideration if we were trying to learn stylistic improvements or other general paraphrase rules from corpora. To explore the feasibility of using a full surface realiser for text simplification, we implemented a module that converts the Stanford dependencies to a DSyntS representation, and used RealPro (Lavoie and Rambow, 1997) to generate sentences. This module is briefly described in §3.2, before we evaluate both approaches in §4.

Summary: To summarise our findings, we find that that the gen-light approach is fairly robust to parsing errors, particularly when the n-best parses are used in an overgenerate-and-rank approach. However, the gen-heavy approach fares less well, since the process of applying transformation rules to an incorrect analysis and then generating with a statistical realiser often leads to garbled output. The gen-heavy approach can be made slightly more robust by using the n-best parses, but the judges in our evaluation still find its word and phrase ordering decisions much less acceptable. Based on our evaluation, we conclude that the preferred solution to regeneration tasks would use a gen-heavy approach for verb features (tense, mood, voice, agreement etc.) and argument ordering, while otherwise reusing word and phrase order from the input.

2 Related work

Text simplification is the process of reducing the grammatical and lexical complexity of a text, while retaining its information content and meaning. The main goal of simplification is to make information more accessible to the large numbers of people with reduced literacy. The National Literacy Trust (http://www.literacytrust.org.uk) estimates that one in six adults in the UK have poor literacy skills; other potential beneficiaries include non-native speakers and children. While there is a large body of evidence that manual text simplification is an effective intervention, there has been relatively little work on automatic simplification.

2.1 Vocabulary, Syntax and Comprehension

There is a large body of research that suggests that there are differences in the way highly skilled and poor readers read. The most striking difference is perhaps at the word level, and people for whom mapping words to meanings requires effort tend to be bad readers (Anderson and Freebody, 1981). However, various studies also highlight the role of syntax in comprehension; for instance, splitting complex sentences into several shorter ones results in better comprehension for less skilled readers (Mason and Kendall, 1979). Similarly, students' reading comprehension has shows to improve when texts have been manually rewritten to make the language more accessible (L'Allier, 1980), or to make the content more transparent (Beck et al., 1991). L'Allier (1980) found that text revision brought low ability readers above the performance level of middle ability readers on the original text and Linderholm et al. (2000) also found that reformulating causal relations for relatively difficult texts had a significant facilitatory effect for readers with low reading skills. However, manually revising texts to fit readers' level of expertise is expensive in terms of both time and money, and there is a need for automatic text simplification systems.

2.2 Automatic Text Simplification

Previous work on automatic syntactic simplification has applied transformation rules to phrasal parse trees. In early work, Chandrasekar and Srinivas (1997) induced simplification rules from a comparison of the structures of the chunked parses of the original and hand-simplified text. The learning algorithm worked by flattening subtrees that were the same on both sides of the rule, replacing identical strings of words with variables and then computing tree→trees transformations to obtain rules in terms of these variables. This work simplified relative clauses, apposition and subordination. The PSET project (Devlin and Tait, 1998; Carroll et al., 1998), which aimed at simplifying news reports for aphasics, followed the approach of Chandrasekar and Srinivas (1997) for syntactic simplification and

focused mainly on lexical simplification (replacing difficult words with easier ones). The PSET project used *WordNet* (Miller et al., 1993) to identify synonyms and the Oxford Psycholinguistic Database (Quinlan, 1992) to determine the relative difficulty of words (Devlin and Tait, 1998).

In more recent work, we have examined syntactic simplification and, in particular, the way syntactic rewrites interact with discourse structure and text cohesion (Siddharthan, 2003; Siddharthan, 2006). This work has spurred subsequent research in using text simplification for second language acquisition (Petersen, 2007) and for increasing access to the web for people with low literacy (Gasperin et al., 2010). However, all these approaches are limited in the kinds of simplification they can perform. For instance, Petersen (2007) found through comparison with manually simplified text that while 87% of split points identified by the Siddharthan (2006) system were correct, these accounted for only 37% of the simplification operations identified in the manually simplified text. Siddharthan (2010) developed a framework that can potentially handle a much wider range of lexico-syntactic simplification operations using transformation rules over type dependency structures, demonstrating their approach using rules to reformulate sentences expressing causality (e.g., " The cause of the explosion was an incendiary device" to " The explosion occurred because of an incendiary device"). In this paper, we build on that work and focus on the issue of robustness in the face of incorrect parser analyses.

2.3 Other text regeneration tasks

Sentence compression is a related research area that aims to shorten sentences for the purpose of summarising the main content. There are similarities between our interest in reformulation and existing work in sentence compression. Sentence compression has usually been addressed in a generative framework, where transformation rules are learnt from parsed corpora of sentences aligned with manually compressed versions. The compression rules learnt are therefore tree-tree transformations (Knight and Marcu, 2000; Galley and McKeown, 2007; Riezler et al., 2003) of some variety. These approaches focus on *deletion* operations, mostly performed low down in the parse tree to remove modifiers. Further they make assumptions about isomorphism between the aligned tree, which means they cannot be readily applied to more complex reformulation operations such as *insertion* and *reordering*. Cohn and Lapata (2009) provide an approach based on Synchronous Tree Substitution Grammar (STSG) that in principle can handle the range of reformulation operations. However, given their focus on sentence compression, they restricted themselves to local transformations near the bottom of the parse tree. Siddharthan (2010) compared different representations and concluded that phrasal parse trees were inadequate for learning complex lexico-syntactic transformation rules and that dependency structures were more suited. Indeed dependency structures are now increasingly popular for other text regeneration tasks, such as sentence fusion (Krahmer et al., 2008; Marsi and Krahmer, 2005; Barzilay and McKeown, 2005).

3 Simplification using typed dependencies

We now summarise REGENT, our system for regenerating text, including two approached to generation: **gen-light** (§3.1) and **gen-heavy** (§3.2).

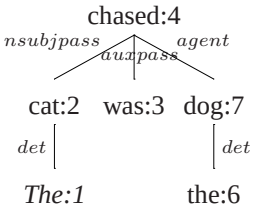
As mentioned before, we use the Stanford parser (De Marneffe et al., 2006) to obtain a typed dependency representation of the input sentence. These are triplets consisting of a relation-type and two arguments. We will use the following sentence to illustrate the process (note that the parser provides word position and part-of-speech tags in addition to dependency relations):

> 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)

To generate, we note that these dependencies represent a tree[1] (we have not shown the punctuation

[1]In fact, the typed dependencies are only 'almost' acyclic. There are a small number of (predictable) relations that introduce cycles .

arc for simplicity):

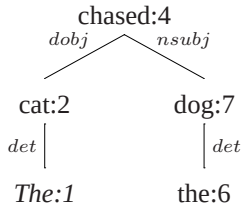


To generate from a dependency tree, we need to know the order in which to process nodes - in general tree traversal will be "inorder"; i.e, left subtrees will be processed before the root and right subtrees after. These are generation decisions that would usually be guided by the type of dependency and statistical preferences for word and phrase order. However, using a gen-light approach, we could simply use the word positions (1–7) from the original sentence, noting that the *agent* relation introduces the word "by".

As typed dependencies can be represented as a flat list, we can write transformation rules quite easily. For instance, a transformation rule to convert the above to active voice would require three deletions and two insertions:

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)

Applying this transformation to the dependency list above creates a new dependency tree:



We can no longer rely on the original word order to determine the order in which to traverse the tree for generation. Now, to generate from this structure, we have two options: gen-light and gen-heavy, summarised below.

3.1 The gen-light approach

If we choose the gen-light approach, our transformation rules, in addition to Deletion and Insertion operations, also need to provide rules for tree traversal order. These only need to be provided for nodes where the transform has reordered subtrees ("??X0", which instantiates to "chased:4" in the trees above). Our rule would thus include:

3. Traversal Order Specifications:
   (a) Node ??X0: [??X3, ??X0, ??X1]

This states that for node ??X0, the traversal order should be subtree ??X3 followed by current node ??X0 followed by subtree ??X1. Using this specification would allow us to traverse the tree using the original word order for nodes with no order specification, and the specified order where a specification exist. In the above instance, this would lead us to generate:

The dog chased the cat.

Our transfer rule is still incomplete and there is one further issue that needs to be addressed – operations to be performed on nodes rather than relations. There are two node-level operation that might be required for sentence reformulation:

Lexical substitution: We still need to ensure number agreement for the verb "chase" (??X0). By changing voice, the verb now has to agree with ??X3 (the dog) rather than ??X1 (the cat). Further the tense of ??X0 was encoded in the auxiliary verb ??X2 (was) that has been deleted from the dependency list. Neither of these matter in the above example, but consider instead a rule for simplifying "The cat is chased by the dogs" to "the dogs chase the cat". We need the transfer rule to encode the lexical substitution required for node ??X0:

4. Lexical substitution:
   (a) Node ??X0: Get tense from ??X2 and number agreement from X3.

Other lexical substitution are easier to specify; for instance to reformulate "*John jumped because David shouted.*" as "*David's shouting caused John to jump*", the following lexical substitution rule is required for node ??Xn representing "shout" that replaces its suffix "ed" with "ing":

Lexical substitution: Node ??Xn: Suffix="ing"

Node deletion: This is an operation that removes a node from the tree. Any subtrees are moved to the parent node. If a root node is deleted, one of the children adopts the rest. By default, the right-most child takes the rest as dependents, but we allow the rule to specify the new parent. In the above example, we want to remove the node ??X2 ("was") (note that deleting a relation does not necessarily remove a node – there might be other nodes connected to ??X2 in the graph). We would like to move these to the node ??X0 ("cause"):

5. Node Deletion:

   (a) Node ??X2: Target=??X0

Node deletion is easily implemented using search and replace on sets of GRs. It is central to any reformulations that alter syntactic categories; for instance, to reformulate "*The cause of X is Y*" as "*Y causes X*", we need to delete the verb "is" and move its dependents to the new verb "causes".

To summarise, the gen-light approach requires transfer rules to specify five 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.

For most reformulations, the CONTEXT and DELETE lists are one and the same, but one can imagine reformulation tasks where extra context needs to be specified to determine whether reformulation is appropriate.

3.2 The gen-heavy approach

The alternative to specifying lists for changes in word/phrase ordering and morphology is to use a formal generator to make these decisions. We use an existing widely used generator RealPro (Lavoie and Rambow, 1997) that uses a typed dependency formalism. For this purpose we have written a convertor that translates the Stanford dependency types into the DSyntS notation required by RealPro. The DSyntS notation differs in two basic ways:

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, auxilliary 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 DSyntS representation created by tranforming the Stanford Dependencies for "the dog chased the cat" is:

```
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:
```

The advantage of the gen-heavy approach is that generation decisions such as ordering and agreement no longer need to be encoded in the transformation rules, making them easier to learn automatically.

3.3 Applying multiple transformation rules

One advantage of using typed dependencies as a representation for applying transformation rules is that we can iteratively apply multiple transformations on the same set of dependency relations. As an illustration, consider:

> The cat was chased by a dog that was barking

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

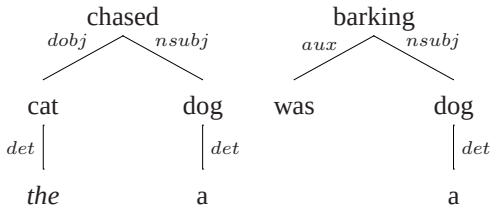
We apply two rules; the first simplifies relative clauses:

1. Match and Delete:
   (a) rcmod(??X0, ??X1)
   (b) nsubj(??X1, ??X0)
2. Insert:
   (a) nsubj(??X1, ??X0)

This rule removes the embedding "rcmod" relation, when there is a subject available for the verb in the relative clause. Then we apply the rule to convert passive to active voice, as described in §3. Following these two rule applications, we are left with the following list of dependencies:

det(cat-2, The-1)
dobj(chased-4, cat-2)
det(dog-7, a-6)
nsubj(chased-4, dog-7)
aux(barking-10, was-9)
nsubj(barking-10, dog-7)

This list now represents two trees with *chased* and *barking* as root nodes:



This generates (using either gen-light or gen-heavy):

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

Note that we employ a postprocessor for generating referring expressions when a noun phrase is repeated. This includes the head noun and either a definite article or a title (e.g., *Mr* or *President*).

3.4 The n-best parse setting

During the development of the system, we found that most of the errors in the output could be traced back to inaccurate parsing. For instance, the top parse for the sentence:

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

identified *which housed* as a relative clause, and *villas* and *blocks* as verbs. This incorrect analysis got simplified (using gen-light) as:

> 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 ask the question, can we use the n-best parses and try to rank the simplified texts in some way? And to what extent can this increase the robustness of the system?

The question arises, how can we evaluate the quality of the generated sentences? Our first attempt calculated n-gram overlap with the original sentence, but this was not found to be useful. Essentially, every transformation of the input sentence reduces ngram overlap between the input and the output, so this method penalises the application of any transforms. This proved to be a problem even for simple transforms such as coordination and subordination that introduce sentence breaks. It proved a bigger problem for embedded constructs such as relative clauses and apposition, and of course the metric is almost meaningless for voice change and other transformations that reorder constituents or change words. Indeed a metric based on comparison with the original would make little sense for text modification applications.

Our final approach was to manually go through the simplifications of the 50 best parses of 100 sentences and identify patterns of infelicities in them. We identified patterns such as:

1. Sentences ending in subject pronouns, prepositions or conjunctions

2. Word repetition (e.g., “is is” or “to to”)

3. Prepositions followed by subject pronouns (e.g., “of he”)

4. Bad sequences of conjunctions and prepositions (e.g., “because but” or “until for”)

Our metric deducted a point for each identified infelicity. In addition we penalised very short sentences (4 words or less) and simplifications that resulted in many fewer words than the original.

In addition to the above penalties, we used the following positive scores:

1. Bigram and trigram overlap with original sentence (as a fraction)

2. The number of sentences in the output (this is to encourage the application of simplification rules)

3. A bonus if the simplification was performed on the top-ranked parse (as this is the most likely to be correct)

In the next section on testing, we report results for system settings using a single parse, and the n-best parses, where we produce n outputs and select the best one according to the criterion described in this section.

4 Evaluation of generation strategies

In this paper we have proposed a framework for complex lexico-syntactic text regeneration. Our system, REGENT, comes with 63 rules for simplifying coordination, subordination, relative clauses, apposition and passive voice. In addition, our system offers two generation options (**gen-light** and **gen-heavy**) in two settings (single and n-best parse).

We emphasise that our purpose in this paper is not to evaluate the simplification rules for their effect on comprehension for different categories of users, but only to test the framework for robustness in the face of parsing errors. We will focus on comparing the four different system settings with respect to how many simplifications have been performed and whether these have been done correctly. Specifically, we will not evaluate whether simplification is found to be useful to different categories of users. With these narrow goals, we report results using:

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

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

Measuring precision as defined above is tricky. As a developer trying to evaluate the framework, the pertinent question is whether the transformation rules have been applied correctly. This however requires knowledge of the transformation rules, which only the developer has. However, we also need external unbiased judgements by testers not involved with the development of the system. These would necessarily conflate issues arising from the quality of the transformation rules with issues arising from the parsing and generation aspects of the system. We present developer test results in §4.1, and an additional evaluation with external testers in §4.2.

4.1 Developer Testing

Our data is six news articles totalling 175 sentences selected as follows: We picked the first two news reports each from the main webpage of three online British news sources (news.bbc.co.uk, guardian.co.uk and thesun.co.uk) at a single time.

We summarise our testing results (with accuracy judged by the developer) in Table 2. In addition, Table 1 provides examples of accurate and inaccurate transformations, as judged by the developer (our judges in §4.2 did not always agree). As Table 2 shows, using the n-best parse setting increases the average number of simplification operations performed by 9 % points and the number of sentences modified by 5 % points. This reduces the average sentence length of the output by around one word. We attribute this improvement to the greater likelihood of a transformation rule matching a dependency parse when multiple parses are considered. More importantly, we also observe an improvement in accuracy from using multiple parses, suggesting that our approach to ranking (§3.4) is valid.

	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	One user of the social network, christonabike, tweeted...
×	One user of the social network tweeted. The network is christonabike...
3	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.
4	Lo and behold the taxpayers subsidised a $30,000 kitchen and he's refusing to give all the details.
√	The taxpayers lo and behold subsidised a $30,000 kitchen. And he is refusing to give all the details.
5	On Thursday, Serbian TV showed footage of the former general wearing a baseball cap and walking slowly as he appeared in court in Belgrade for the first time.
√	Serbian TV showed footage of the former general wearing a baseball cap and slowly walking on Thursday. This is as he appeared in court in Belgrade for the first time.

Table 1: Examples of automatic reformulations.

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

Table 2: Test results for four configurations of the system: **gen-l**ight and **gen-h**eavy 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.

Manual inspection of the mistakes reveals that for the gen-light approach with n-best parses, these are mostly due to relative clause attachment errors by the parser – these result in incorrect simplification, but no disfluency in the output (cf. Ex 2 in Table 1).

The gen-heavy approach makes many more errors; these are usually due to mis-parses (cf. Ex 1 and 3 in Table 1; in 3, while the word order is fine, the parser has incorrectly detected a reduced relative clause, and RealPro has placed this within commas). In addition, the gen-heavy approach often results in different phrase orders that might be harder to read, for instance in Ex 4 and 5. These have been treated as accurate in this evaluation.

4.2 Acceptability measurements

In the previous section, the developer was testing whether the rules have been applied accurately. This is different from evaluating the acceptability of the output. We selected 50 sentences from our test set at random and asked two native speakers to label each of the four reformulations (gen-light and gen-heavy in single and 50-best parse settings) as either acceptable or unacceptable. The judges were shown both the original sentence and reformulations, without being provided information about system settings. We found quite a low pair-wise agreement ($kappa < 0.55$) between the two judges and also with the developer's judgements for these sentences (Table 3). Table 4 shows the acceptability results are lower than the developer's assessments in the previous section, particularly for the gen-heavy approach. The two judges deemed sentences unacceptable for a variety of reasons that were not penalised by the developer in his testing (e.g., disfluencies that were carried over from the input, incorrect capitalisation, lack of punctuation, bad sentence order, etc.).

In addition, the judges also deemed examples such as 5 in Table 1 unacceptable because of the copula being in present tense. The "this is" construct is introduced by the transformation rules for subordination; however, this fact is only known to the developer (who thus deemed the transformation accu-

Judge 1	Judge 2	κ	% Agreement
A	B	.55	78%
A	Developer	.52	79%
B	Developer	.32	68%

Table 3: Pairwise agreement on acceptability

System	% Acceptable			
	J1	J2	Developer	Majority
gen-l/1 parse	.59	.66	.79	.69
gen-l/50 parses	.62	.69	.86	.78
gen-h/1 parse	.19	.40	.62	.40
gen-h/50 parses	.20	.45	.71	.43

Table 4: 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.

rate) and not to the two judges (who thus deemed the output unacceptable).

We believe that these two evaluation provide a good indication of the performance of our system and its different settings, as well as the quality of the transformation rules. The main conclusion that we draw from these tests is that users can be quite intolerant towards suboptimal word ordering, and that using an off-the shelf sentence realiser is not a good option for text regeneration tasks, unless it can reuse ordering information from the input in some way.

5 Conclusions and future work

We have presented a system for text simplification based on applying transformation rules to typed dependencies. The main contribution of this paper is to demonstrate that the robustness of the system to parsing errors can be improved by using the n-best dependency parses in a overgenerate-and-rank approach. In addition, we explore the question of whether an existing surface realiser can be used for text regeneration tasks. We find that this approach is brittle, and 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 in the input sentence, using deep generation only for verb features (mood, tense, voice, etc.), number agreement and argument order.

Acknowledgements

This work was supported by the Economic and Social Research Council (Grant Number RES-000-22-3272).

References

Richard Anderson and Peter Freebody. 1981. Vocabulary knowledge. In John Guthrie, editor, *Comprehension and Teaching: Research Reviews*, pages 77–117. International Reading Association, Newark, DE.

Richard Anderson. 1981. A proposal to continue a center for the study of reading. Technical Report 487, University of Illinois, Center for the Study of Reading, Urbana-Champaign.

Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *HLT-NAACL 2003: Main Proceedings*, pages 16–23.

Regina Barzilay and Kathleen McKeown. 2005. Sentence fusion for multidocument news summarization. *Computational Linguistics*, 31(3):297–328.

Isabel L. Beck, Margaret G. McKeown, Gale M. Sinatra, and Jane A. Loxterman. 1991. Revising social studies text from a text-processing perspective: Evidence of improved comprehensibility. *Reading Research Quarterly*, pages 251–276.

John Carroll, Guido Minnen, Yvonne Canning, Siobhan Devlin, and John Tait. 1998. Practical simplification of English newspaper text to assist aphasic readers. In *Proceedings of AAAI98 Workshop on Integrating Artificial Intelligence and Assistive Technology*, pages 7–10, Madison, Wisconsin.

Raman Chandrasekar and Bangalore Srinivas. 1997. Automatic induction of rules for text simplification. *Knowledge-Based Systems*, 10:183–190.

Raman Chandrasekar, Christine Doran, and Bangalore Srinivas. 1996. Motivations and methods for text simplification. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING '96)*, pages 1041–1044, Copenhagen, Denmark.

Trevor Cohn and Mirella Lapata. 2009. Sentence compression as tree transduction. *Journal of Artificial Intelligence Research*, 34(1):637–674.

Trevor Cohn, Chris Callison-Burch, and Mirella Lapata. 2008. Constructing corpora for the development and evaluation of paraphrase systems. *Computational Linguistics*, 34(4):597–614.

Meredyth Daneman and Patricia Carpenter. 1980. Individual differences in working memory and reading. *Journal of Verbal Learning and Verbal Behavior*, 19:450–466.

Marie-Catherine De Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454. Citeseer.

Siobhan Devlin and John Tait. 1998. The use of a psycholinguistic database in the simplification of text for aphasic readers. In J. Nerbonne, editor, *Linguistic Databases*, pages 161–173. CSLI Publications, Stanford, California.

Michel Galley and Kathleen McKeown. 2007. Lexicalized Markov grammars for sentence compression. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 180–187, Rochester, New York, April. Association for Computational Linguistics.

Caroline Gasperin, Erick Maziero, and Sandra Aluísio. 2010. Challenging choices for text simplification. *Computational Processing of the Portuguese Language*, pages 40–50.

Ali Ibrahim, Boris Katz, and Jimmy Lin. 2003. Extracting paraphrases from aligned corpora. In *Proceedings of The Second International Workshop on Paraphrasing*.

Nobuhiro Kaji, Daisuke Kawahara, Sadao Kurohash, and Satoshi Sato. 2002. Verb paraphrase based on case frame alignment. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL'02)*, pages 215–222, Philadelphia, USA.

Kevin Knight and Daniel Marcu. 2000. Statistics-based summarization — step one: Sentence compression. In *Proceeding of The American Association for Artificial Intelligence Conference (AAAI-2000)*, pages 703–710.

Emiel Krahmer, Erwin Marsi, and Paul van Pelt. 2008. Query-based sentence fusion is better defined and leads to more preferred results than generic sentence fusion. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, pages 193–196. Association for Computational Linguistics.

James J. L'Allier. 1980. *An evaluation study of a computer-based lesson that adjusts reading level by monitoring on task reader characteristics*. Ph.D. thesis, University of Minnesota, Minneapolis, MN.

Benoit Lavoie and Owen Rambow. 1997. A fast and portable realizer for text generation systems. In *Proceedings of the fifth conference on Applied natural language processing*, pages 265–268. Association for Computational Linguistics.

Tracy Linderholm, Michelle G. Everson, Paul van den Broek, Maureen Mischinski, Alex Crittenden, and J. Samuels. 2000. Effects of Causal Text Revisions on More-and Less-Skilled Readers' Comprehension of Easy and Difficult Texts. *Cognition and Instruction*, 18(4):525–556.

Erwin Marsi and Emiel Krahmer. 2005. Explorations in sentence fusion. In *Proceedings of the European Workshop on Natural Language Generation*, pages 109–117.

Jana Mason and Janet Kendall. 1979. Facilitating reading comprehension through text structure manipulation. *Alberta Journal of Medical Psychology*, 24:68–76.

George A. Miller, Richard Beckwith, Christiane D. Fellbaum, Derek Gross, and Katherine Miller. 1993. Five Papers on WordNet. Technical report, Princeton University, Princeton, N.J.

Sarah E. Petersen. 2007. *Natural language processing tools for reading level assessment and text simplification for bilingual education*. Ph.D. thesis, University of Washington, Seattle, WA.

Stephen P. Quigley and Peter V. Paul. 1984. *Language and Deafness*. College-Hill Press, San Diego, California.

Philip Quinlan. 1992. *The Oxford Psycholinguistic Database*. Oxford University Press, U.K.

Stefan Riezler, Tracy H. King, Richard Crouch, and Annie Zaenen. 2003. Statistical sentence condensation using ambiguity packing and stochastic disambiguation methods for lexical-functional grammar. In *Proceedings of the Human Language Technology Conference and the 3rd Meeting of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL'03)*, Edmonton, Canada.

Advaith Siddharthan and Napoleon Katsos. 2010. Reformulating discourse connectives for non-expert readers. In *Proceedings of the 11th Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT 2010)*, Los Angeles, CA.

Advaith Siddharthan. 2003. Preserving discourse structure when simplifying text. In *Proceedings of the European Natural Language Generation Workshop (ENLG), 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL'03)*, pages 103–110, Budapest, Hungary.

Advaith Siddharthan. 2006. Syntactic simplification and text cohesion. *Research on Language and Computation*, 4(1):77–109.

Advaith Siddharthan. 2010. Complex lexico-syntactic reformulation of sentences using typed dependency representations. In *Proceedings of the 6th International Natural Language Generation Conference (INLG 2010)*, pages 125–133. Dublin, Ireland.