A Monolingual Tree-based Translation Model for Sentence Simplification*

Zhemin Zhu[1]
Department of Computer Science
Technische Universität Darmstadt
[1]http://www.ukp.tu-darmstadt.de

Delphine Bernhard[2]
LIMSI-CNRS
[2]delphine.bernhard@limsi.fr

Iryna Gurevych[1]
Department of Computer Science
Technische Universität Darmstadt

Abstract

In this paper, we consider sentence simplification as a special form of translation with the complex sentence as the source and the simple sentence as the target. We propose a Tree-based Simplification Model (TSM), which, to our knowledge, is the first statistical simplification model covering splitting, dropping, reordering and substitution integrally. We also describe an efficient method to train our model with a large-scale parallel dataset obtained from the Wikipedia and Simple Wikipedia. The evaluation shows that our model achieves better readability scores than a set of baseline systems.

1 Introduction

Sentence simplification transforms long and difficult sentences into shorter and more readable ones. This helps humans read texts more easily and faster. Reading assistance is thus an important application of sentence simplification, especially for people with reading disabilities (Carroll et al., 1999; Inui et al., 2003), low-literacy readers (Watanabe et al., 2009), or non-native speakers (Siddharthan, 2002).

Not only human readers but also NLP applications can benefit from sentence simplification. The original motivation for sentence simplification is using it as a preprocessor to facilitate parsing or translation tasks (Chandrasekar et al., 1996). Complex sentences are considered as stumbling blocks for such systems. More recently, sentence simplification has also been shown helpful for summarization (Knight and Marcu, 2000), sentence fusion (Filippova and Strube, 2008b), semantic role labeling (Vickrey and Koller, 2008), question generation (Heilman and Smith, 2009), paraphrase generation (Zhao et al., 2009) and biomedical information extraction (Jonnalagadda and Gonzalez, 2009).

At sentence level, reading difficulty stems either from lexical or syntactic complexity. Sentence simplification can therefore be classified into two types: *lexical simplification* and *syntactic simplification* (Carroll et al., 1999). These two types of simplification can be further implemented by a set of simplification operations. *Splitting*, *dropping*, *reordering*, and *substitution* are widely accepted as important simplification operations. The splitting operation splits a long sentence into several shorter sentences to decrease the complexity of the long sentence. The dropping operation further removes unimportant parts of a sentence to make it more concise. The reordering operation interchanges the order of the split sentences (Siddharthan, 2006) or parts in a sentence (Watanabe et al., 2009). Finally, the substitution operation replaces difficult phrases or words with their simpler synonyms.

In most cases, different simplification operations happen simultaneously. It is therefore necessary to consider the simplification process as a combination of different operations and treat them as a whole. However, most of the existing models only consider one of these operations. Siddharthan (2006) and Petersen and Ostendorf (2007) focus on sentence splitting, while sentence compression systems (Filippova and Strube, 2008a) mainly use the dropping operation. As far as lexical simplification is concerned, word substitution is usually done by selecting simpler synonyms from Wordnet based on word frequency (Carroll et al., 1999).

In this paper, we propose a sentence simplification model by tree transformation which is based

* This work has been supported by the Emmy Noether Program of the German Research Foundation (DFG) under the grant No. GU 798/3-1, and by the Volkswagen Foundation as part of the Lichtenberg-Professorship Program under the grant No. I/82806.

Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pages 1353–1361,
Beijing, August 2010

on techniques from statistical machine translation (SMT) (Yamada and Knight, 2001; Yamada and Knight, 2002; Graehl et al., 2008). Our model integrally covers splitting, dropping, reordering and phrase/word substitution. The parameters of our model can be efficiently learned from complex-simple parallel datasets. The transformation from a complex sentence to a simple sentence is conducted by applying a sequence of simplification operations. An expectation maximization (EM) algorithm is used to iteratively train our model. We also propose a method based on monolingual word mapping which speeds up the training process significantly. Finally, a decoder is designed to generate the simplified sentences using a greedy strategy and integrates language models.

In order to train our model, we further compile a large-scale complex-simple parallel dataset (PWKP) from Simple English Wikipedia[1] and English Wikipedia[2], as such datasets are rare.

We organize the remainder of the paper as follows: Section 2 describes the PWKP dataset. Section 3 presents our TSM model. Sections 4 and 5 are devoted to training and decoding, respectively. Section 6 details the evaluation. The conclusions follow in the final section.

2 Wikipedia Dataset: PWKP

We collected a paired dataset from the English Wikipedia and Simple English Wikipedia. The targeted audience of Simple Wikipedia includes "children and adults who are learning English language". The authors are requested to "use easy words and short sentences" to compose articles. We processed the dataset as follows:

Article Pairing 65,133 articles from Simple Wikipedia[3] and Wikipedia[4] were paired by following the "language link" using the dump files in Wikimedia.[5] Administration articles were further removed.

Plain Text Extraction We use JWPL (Zesch et al., 2008) to extract plain texts from Wikipedia articles by removing specific Wiki tags.

Pre-processing including sentence boundary detection and tokenization with the Stanford Parser package (Klein and Manning, 2003), and lemmatization with the TreeTagger (Schmid, 1994).

Monolingual Sentence Alignment As we need a parallel dataset aligned at the sentence level, we further applied monolingual sentence alignment on the article pairs. In order to achieve the best sentence alignment on our dataset, we tested three similarity measures: (i) sentence-level TF*IDF (Nelken and Shieber, 2006), (ii) word overlap (Barzilay and Elhadad, 2003) and (iii) word-based maximum edit distance (MED) (Levenshtein, 1966) with costs of insertion, deletion and substitution set to 1. To evaluate their performance we manually annotated 120 sentence pairs from the article pairs. Tab. 1 reports the precision and recall of these three measures. We manually adjusted the similarity threshold to obtain a recall value as close as possible to 55.8% which was previously adopted by Nelken and Shieber (2006).

Similarity	Precision	Recall
TF*IDF	91.3%	55.4%
Word Overlap	50.5%	55.1%
MED	13.9%	54.7%

Table 1: Monolingual Sentence Alignment

The results in Tab. 1 show that sentence-level TF*IDF clearly outperforms the other two measures, which is consistent with the results reported by Nelken and Shieber (2006). We henceforth chose sentence-level TF*IDF to align our dataset.

As shown in Tab. 2, PWKP contains more than 108k sentence pairs. The sentences from Wikipedia and Simple Wikipedia are considered as "complex" and "simple" respectively. Both the average sentence length and average token length in Simple Wikipedia are shorter than those in Wikipedia, which is in compliance with the purpose of Simple Wikipedia.

Avg. Sen. Len		Avg. Tok. Len		#Sen.Pairs
complex	simple	complex	simple	-
25.01	20.87	5.06	4.89	108,016

Table 2: Statistics for the PWKP dataset

In order to account for sentence splitting, we allow 1 to n sentence alignment to map one complex sentence to several simple sentences. We first perform 1 to 1 mapping with sentence-level TF*IDF and then combine the pairs with the same complex sentence and adjacent simple sentences.

3 The Simplification Model: TSM

We apply the following simplification operations to the parse tree of a complex sentence: splitting,

[1] http://simple.wikipedia.org
[2] http://en.wikipedia.org
[3] As of Aug 17th, 2009
[4] As of Aug 22nd, 2009
[5] http://download.wikimedia.org

dropping, reordering and substitution. In this section, we use a running example to illustrate this process. c is the complex sentence to be simplified in our example. Fig. 1 shows the parse tree of c (we skip the POS level).

c: *August was the sixth month in the ancient Roman calendar which started in 735BC.*

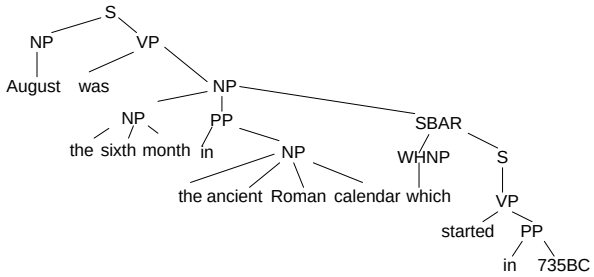

Figure 1: Parse Tree of c

3.1 Splitting

The first operation is sentence splitting, which we further decompose into two subtasks: (i) *segmentation*, which decides where and whether to split a sentence and (ii) *completion*, which makes the new split sentences complete.

First, we decide where we can split a sentence. In our model, the splitting point is judged by the syntactic constituent of the split boundary word in the complex sentence. The decision whether a sentence should be split is based on the length of the complex sentence. The features used in the *segmentation* step are shown in Tab. 3.

Word	Constituent	iLength	isSplit	Prob.
"which"	SBAR	1	true	0.0016
"which"	SBAR	1	false	0.9984
"which"	SBAR	2	true	0.0835
"which"	SBAR	2	false	0.9165

Table 3: Segmentation Feature Table (SFT)

Actually, we do not use the direct constituent of a word in the parse tree. In our example, the direct constituent of the word "which" is "WHNP". Instead, we use Alg. 1 to calculate the constituent of a word. Alg. 1 returns "SBAR" as the adjusted constituent for "which". Moreover, directly using the length of the complex sentence is affected by the data sparseness problem. Instead, we use $\mathrm{iLength}$ as the feature which is calculated as $\mathrm{iLength} = \mathrm{ceiling}(\frac{\mathrm{comLength}}{\mathrm{avgSimLength}})$, where $\mathrm{comLength}$ is the length of the complex sentence and $\mathrm{avgSimLength}$ is the average length of simple sentences in the training dataset. The "Prob." column shows the probabilities obtained after training on our dataset.

Algorithm 1 adjustConstituent($word$, $tree$)

```
constituent ← word.father;
father ← constituent.father;
while father ≠ NULL AND constituent is the most
left child of father do
    constituent ← father;
    father ← father.father;
end while
return constituent;
```

In our model, one complex sentence can be split into two or more sentences. Since many splitting operations are possible, we need to select the most likely one. The probability of a segmentation operation is calculated as:

$$P(seg|c) = \prod_{w:c} SFT(w|c) \quad (1)$$

where w is a word in the complex sentence c and $SFT(w|c)$ is the probability of the word w in the Segmentation Feature Table (SFT); Fig. 2 shows a possible segmentation result of our example.

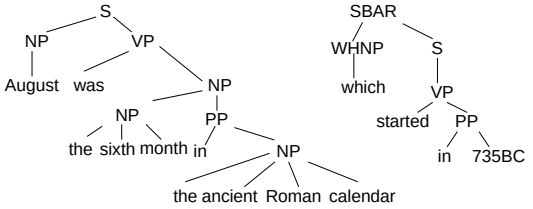

Figure 2: Segmentation

The second step is *completion*. In this step, we try to make the split sentences complete and grammatical. In our example, to make the second sentence "which started in 735BC" complete and grammatical we should first drop the border word "which" and then copy the dependent NP "the ancient Roman calendar" to the left of "started" to obtain the complete sentence "the ancient Roman calendar started in 735BC". In our model, whether the border word should be dropped or retained depends on two features of the border word: the direct constituent of the word and the word itself, as shown in Tab. 4.

Const.	Word	isDropped	Prob.
WHNP	which	True	1.0
WHNP	which	False	Prob.Min

Table 4: Border Drop Feature Table (BDFT)

In order to copy the necessary parts to complete the new sentences, we must decide which parts should be copied and where to put these parts in the new sentences. In our model, this is judged by two features: the dependency relation and the constituent. We use the Stanford Parser for parsing the dependencies. In our example, the de-

pendency relation between "calendar" in the complex sentence and the verb "started" in the second split sentence is "gov_nsubj".[6] The direct constituent of "started" is "VP" and the word "calendar" should be put on the "left" of "started", see Tab. 5.

Dep.	Const.	isCopied	Pos.	Prob.
gov_nsubj	VP(VBD)	True	left	0.9000
gov_nsubj	VP(VBD)	True	right	0.0994
gov_nsubj	VP(VBD)	False	-	0.0006

Table 5: Copy Feature Table (CFT)

For dependent NPs, we copy the whole NP phrase rather than only the head noun.[7] In our example, we copy the whole NP phrase "the ancient Roman calendar" to the new position rather than only the word "calendar". The probability of a completion operation can be calculated as

$$P(com|seg) = \prod_{bw:s} BDFT(bw|s) \prod_{w:s} \prod_{dep:w} CFT(dep).$$

where s are the split sentences, bw is a border word in s, w is a word in s, dep is a dependency of w which is out of the scope of s. Fig. 3 shows the most likely result of the completion operation for our example.

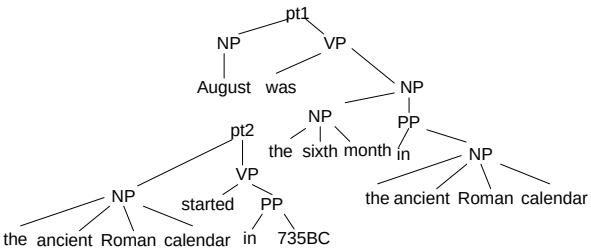

Figure 3: Completion

3.2 Dropping and Reordering

We first apply dropping and then reordering to each non-terminal node in the parse tree from top to bottom. We use the same features for both dropping and reordering: the node's direct constituent and its children's constituents pattern, see Tab. 6 and Tab. 7.

Constituent	Children	Drop	Prob.
NP	DT_JJ_NNP_NN	1101	7.66E-4
NP	DT_JJ_NNP_NN	0001	1.26E-7

Table 6: Dropping Feature Table (DFT)

[6]With Stanford Parser, "which" is a referent of "calender" and the nsubj of "started". "calender" thus can be considered to be the nsubj of "started" with "started" as the governor.

[7]The copied NP phrase can be further simplified in the following steps.

Constituent	Children	Reorder	Prob.
NP	DT_JJ_NN	012	0.8303
NP	DT_JJ_NN	210	0.0039

Table 7: Reordering Feature Table (RFT)

The bits '1' and '0' in the "Drop" column indicate whether the corresponding constituent is retained or dropped. The number in the "Reorder" column represents the new order for the children. The probabilities of the dropping and reordering operations can be calculated as Equ. 2 and Equ. 3.

$$P(dp|node) = DFT(node) \quad (2)$$

$$P(ro|node) = RFT(node) \quad (3)$$

In our example, one of the possible results is dropping the NNP "Roman", as shown in Fig. 4.

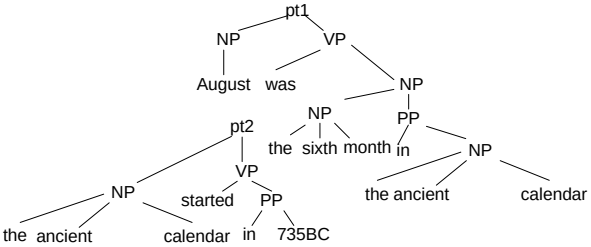

Figure 4: Dropping & Reordering

3.3 Substitution

3.3.1 Word Substitution

Word substitution only happens on the terminal nodes of the parse tree. In our model, the conditioning features include the original word and the substitution. The substitution for a word can be another word or a multi-word expression (see Tab. 8). The probability of a word substitution operation can be calculated as $P(sub|w) = SubFT(Substitution|Origin)$.

Origin	Substitution	Prob.
ancient	ancient	0.963
ancient	old	0.0183
ancient	than transport	1.83E-102
old	ancient	0.005

Table 8: Substitution Feature Table (SubFT)

3.3.2 Phrase Substitution

Phrase substitution happens on the non-terminal nodes and uses the same conditioning features as word substitution. The "Origin" consists of the leaves of the subtree rooted at the node. When we apply phrase substitution on a non-terminal node, then any simplification operation (including dropping, reordering and substitution) cannot happen on its descendants any more

because when a node has been replaced then its descendants are no longer existing. Therefore, for each non-terminal node we must decide whether a substitution should take place at this node or at its descendants. We perform substitution for a non-terminal $node$ if the following constraint is met:

$$Max(SubFT(*|node)) \geq \prod_{ch:node} Max(SubFT(*|ch)).$$

where ch is a child of the $node$. "$*$" can be any substitution in the SubFT. The probability of the phrase substitution is calculated as $P(sub|node) = SubFT(Substitution|Origin)$. Fig. 5 shows one of the possible substitution results for our example where "ancient" is replaced by "old".

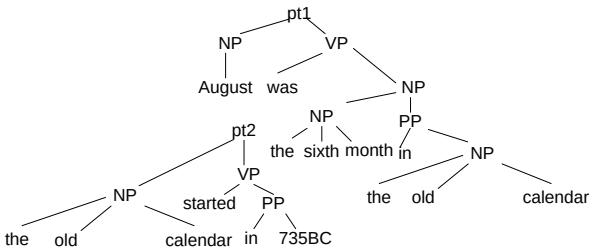

Figure 5: Substitution

As a result of all the simplification operations, we obtain the following two sentences: $s1 = Str(pt1)$="*August was the sixth month in the old calendar.*" and $s2 = Str(pt2)$="*The old calendar started in 735BC.*"

3.4 The Probabilistic Model

Our model can be formalized as a direct translation model from complex to simple $P(s|c)$ multiplied by a language model $P(s)$ as shown in Equ. 4.

$$s = \underset{s}{\operatorname{argmax}} P(s|c)P(s) \quad (4)$$

We combine the parts described in the previous sections to get the direct translation model:

$$P(s|c) = \sum_{\theta:Str(\theta(c))=s} (P(seg|c)P(com|seg) \quad (5)$$
$$\prod_{node} P(dp|node)P(ro|node)P(sub|node)$$
$$\prod_{w} (sub|w)).$$

where θ is a sequence of simplification operations and $Str(\theta(c))$ corresponds to the leaves of a simplified tree. There can be many sequences of operations that result in the same simplified sentence and we sum up all of their probabilities.

4 Training

In this section, we describe how we train the probabilities in the tables. Following the work of Yamada and Knight (2001), we train our model by maximizing $P(s|c)$ over the training corpus with the EM algorithm described in Alg. 2, using a constructed graph structure. We develop the Training Tree (Fig. 6) to calculate $P(s|c)$. $P(s|c)$ is equal to the inside probability of the root in the Training Tree. Alg. 3 and Alg. 4 are used to calculate the inside and outside probabilities. We refer readers to Yamada and Knight (2001) for more details.

Algorithm 2 EM Training ($dataset$)

```
Initialize all probability tables using the uniform distribution;
for several iterations do
    reset all cnt = 0;
    for each sentence pair < c, s > in dataset do
        tt = buildTrainingTree(< c, s >);
        calcInsideProb(tt);
        calcOutsideProb(tt);
        update cnt for each conditioning feature in each
        node of tt: cnt = cnt + node.insideProb *
        node.outsideProb/root.insideProb;
    end for
    updateProbability();
end for
```

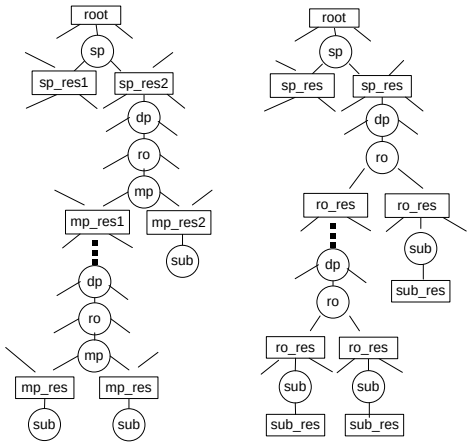

Figure 6: Training Tree (Left) and Decoding Tree (Right)

We illustrate the construction of the training tree with our running example. There are two kinds of nodes in the training tree: data nodes in rectangles and operation nodes in circles. Data nodes contain data and operation nodes execute operations. The training is a supervised learning

process with the parse tree of c as input and the two strings $s1$ and $s2$ as the desired output. $root$ stores the parse tree of c and also $s1$ and $s2$. sp, ro, mp and sub are splitting, reordering, mapping and substitution operations. sp_res and mp_res store the results of sp and mp. In our example, sp splits the parse tree into two parse trees $pt1$ and $pt2$ (Fig. 3). sp_res1 contains $pt1$ and $s1$. sp_res2 contains $pt2$ and $s2$. Then dp, ro and mp are iteratively applied to each non-terminal node at each level of $pt1$ and $pt2$ from top to down. This process continues until the terminal nodes are reached or is stopped by a sub node. The function of mp operation is similar to the word mapping operation in the string-based machine translation. It maps substrings in the complex sentence which are dominated by the children of the current node to proper substrings in the simple sentences.

Speeding Up The example above is only one of the possible paths. We try all of the promising paths in training. Promising paths are the paths which are likely to succeed in transforming the parse tree of c into $s1$ and $s2$. We select the promising candidates using monolingual word mapping as shown in Fig. 7. In this example, only the word "which" can be a promising candidate for splitting. We can select the promising candidates for the dropping, reordering and mapping operations similarly. With this improvement, we can train on the PWKP dataset within 1 hour excluding the parsing time taken by the Stanford Parser.

We initialize the probabilities with the uniform distribution. The binary features, such as SFT and BDFT, are assigned the initial value of 0.5. For DFT and RFT, the initial probability is $\frac{1}{N!}$, where N is the number of the children. CFT is initialized as 0.25. SubFT is initialized as 1.0 for any substitution at the first iteration. After each iteration, the $updateProbability$ function recalculates these probabilities based on the cnt for each feature.

Algorithm 3 calcInsideProb (TrainingTree tt)

for each $node$ from $level = N$ to $root$ of tt **do**
  if $node$ is a sub node **then**
    $node.insideProb$ = $P(sub|node)$;
  else if $node$ is a mp OR sp node **then**
    $node.insideProb$ = $\prod_{child} child.insideProb$;
  else
    $node.insideProb$ = $\sum_{child} child.insideProb$;
  end if
end for

Algorithm 4 calcOutsideProb (TrainingTree tt)

for each $node$ from $root$ to $level = N$ of tt **do**
  if $node$ is the $root$ **then**
    $node.outsideProb$ = 1.0;
  else if $node$ is a sp_res OR mp_res node **then**
    {COMMENT: $father$ are the fathers of the current $node$, $sibling$ are the children of $father$ excluding the current $node$}
    $node.outsideProb = \sum_{father} father.outsideProb * \prod_{sibling} sibling.insideProb$;
  else if $node$ is a mp node **then**
    $node.outsideProb$ = $father.outsideProb * 1.0$;
  else if $node$ is a sp, ro, dp or sub node **then**
    $node.outsideProb = father.outsideProb * P(sp$ or ro or dp or $sub|node)$;
  end if
end for

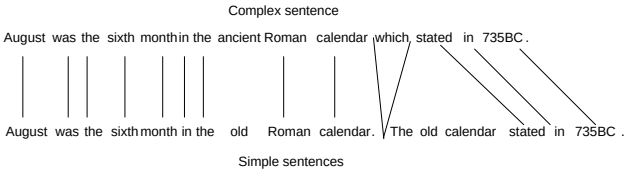

Figure 7: Monolingual Word Mapping

5 Decoding

For decoding, we construct the decoding tree (Fig. 6) similarly to the construction of the training tree. The decoding tree does not have mp operations and there can be more than one sub nodes attached to a single ro_res. The $root$ contains the parse tree of the complex sentence. Due to space limitations, we cannot provide all the details of the decoder.

We calculate the inside probability and outside probability for each node in the decoding tree. When we simplify a complex sentence, we start from the root and greedily select the branch with the highest outside probability. For the substitution operation, we also integrate a trigram language model to make the generated sentences more fluent. We train the language model with SRILM (Stolcke, 2002). All the articles from the Simple Wikipedia are used as the training corpus, amounting to about 54 MB.

6 Evaluation

Our evaluation dataset consists of 100 complex sentences and 131 parallel simple sentences from PWKP. They have not been used for training. Four baseline systems are compared in our evaluation. The first is Moses which is a state of the art SMT system widely used as a baseline in MT community. Obviously, the purpose of Moses is cross-lingual translation rather than monolin-

gual simplification. The goal of our comparison is therefore to assess how well a standard SMT system may perform simplification when fed with a proper training dataset. We train Moses with the same part of PWKP as our model. The second baseline system is a sentence compression system (Filippova and Strube, 2008a) whose demo system is available online.[8] As the compression system can only perform dropping, we further extend it to our third and fourth baseline systems, in order to make a reasonable comparison. In our third baseline system, we substitute the words in the output of the compression system with their simpler synonyms. This is done by looking up the synonyms in Wordnet and selecting the most frequent synonym for replacement. The word frequency is counted using the articles from Simple Wikipedia. The fourth system performs sentence splitting on the output of the third system. This is simply done by splitting the sentences at "and", "or", "but", "which", "who" and "that", and discarding the border words. In total, there are 5 systems in our evaluation: *Moses*, the MT system; *C*, the compression system; *CS*, the compression+substitution system; *CSS*, the compression+substitution+split system; *TSM*, our model. We also provide evaluation measures for the sentences in the evaluation dataset: *CW*: complex sentences from Normal Wikipedia and *SW*: parallel simple sentences from Simple Wikipedia.

6.1 Basic Statistics and Examples

The first three columns in Tab. 9 present the basic statistics for the evaluation sentences and the output of the five systems. $tokenLen$ is the average length of tokens which may roughly reflect the lexical difficulty. TSM achieves an average token length which is the same as the Simple Wikipedia (SW). $senLen$ is the average number of tokens in one sentence, which may roughly reflect the syntactic complexity. Both TSM and CSS produce shorter sentences than SW. Moses is very close to CW. $\#sen$ gives the number of sentences. Moses, C and CS cannot split sentences and thus produce about the same number of sentences as available in CW.

Here are two example results obtained with our TSM system.

Example 1. *CW*: "Genetic engineering has expanded the genes available to breeders to utilize in creating desired germlines for new crops." *SW*: "New plants were created with genetic engineering." *TSM*: "Engineering has expanded the genes available to breeders to use in making germlines for new crops."

Example 2. *CW*: "An umbrella term is a word that provides a superset or grouping of related concepts, also called a hypernym." *SW*: "An umbrella term is a word that provides a superset or grouping of related concepts." *TSM*: "An umbrella term is a word. A word provides a superset of related concepts, called a hypernym."

In the first example, both substitution and dropping happen. TSM replaces "utilize" and "creating" with "use" and "making". "Genetic" is dropped. In the second example, the complex sentence is split and "also" is dropped.

6.2 Translation Assessment

In this part of the evaluation, we use traditional measures used for evaluating MT systems. Tab. 9 shows the BLEU and NIST scores. We use "mteval-v11b.pl"[9] as the evaluation tool. CW and SW are used respectively as source and reference sentences. TSM obtains a very high BLEU score (0.38) but not as high as Moses (0.55). However, the original complex sentences (CW) from Normal Wikipedia get a rather high BLEU (0.50), when compared to the simple sentences. We also find that most of the sentences generated by Moses are exactly the same as those in CW: this shows that Moses only performs few modifications to the original complex sentences. This is confirmed by MT evaluation measures: if we set CW as both source and reference, the BLEU score obtained by Moses is 0.78. TSM gets 0.55 in the same setting which is significantly smaller than Moses and demonstrates that TSM is able to generate simplifications with a greater amount of variation from the original sentence. As shown in the "#Same" column of Tab. 9, 25 sentences generated by Moses are exactly identical to the complex sentences, while the number for TSM is 2 which is closer to SW. It is however not clear how well BLEU and NIST discriminate simplification systems. As discussed in Jurafsky and Martin (2008), "BLEU does poorly at comparing systems with radically different architectures and is most appropriate when evaluating incremental changes with similar architectures." In our case, TSM and CSS can be considered as having similar architectures as both of them can do splitting, dropping

[8] http://212.126.215.106/compression/

[9] http://www.statmt.org/moses/

	TokLen	SenLen	#Sen	BLEU	NIST	#Same	Flesch	Lix(Grade)	OOV%	PPL
CW	4.95	27.81	100	0.50	6.89	100	49.1	53.0 (10)	52.9	384
SW	4.76	17.86	131	1.00	10.98	3	60.4 (PE)	44.1 (8)	50.7	179
Moses	4.81	26.08	100	**0.55**	**7.47**	25	54.8	48.1 (9)	52.0	363
C	4.98	18.02	103	0.28	5.37	1	56.2	45.9 (8)	51.7	481
CS	4.90	18.11	103	0.19	4.51	0	59.1	45.1 (8)	**49.5**	616
CSS	4.98	**10.20**	**182**	0.18	4.42	0	65.5 (PE)	38.3 (6)	53.4	581
TSM	**4.76**	13.57	180	0.38	6.21	**2**	**67.4 (PE)**	**36.7 (5)**	50.8	**353**

Table 9: Evaluation

and substitution. But Moses mostly cannot split and drop. We may conclude that TSM and Moses have different architectures and BLEU or NIST is not suitable for comparing them. Here is an example to illustrate this: *(CW):* "Almost as soon as he leaves, Annius and the guard Publius arrive to **escort** Vitellia to Titus, who has now chosen her as his empress." *(SW):* "Almost as soon as he leaves, Annius and the guard Publius arrive to **take** Vitellia to Titus, who has now chosen her as his empress." *(Moses):* The same as (SW). *(TSM):* "Annius and the guard Publius arrive to **take** Vitellia to Titus. Titus has now chosen her as his empress." In this example, *Moses* generates an exactly identical sentence to *SW*, thus the BLUE and NIST scores of *Moses* is the highest. *TSM* simplifies the complex sentence by dropping, splitting and substitution, which results in two sentences that are quite different from the *SW* sentence and thus gets lower BLUE and NIST scores. Nevertheless, the sentences generated by *TSM* seem better than *Moses* in terms of simplification.

6.3 Readability Assessment

Intuitively, readability scores should be suitable metrics for simplification systems. We use the Linux "style" command to calculate the Flesch and Lix readability scores. The results are presented in Tab. 9. "PE" in the Flesch column stands for "Plain English" and the "Grade" in Lix represents the school year. TSM achieves significantly better scores than Moses which has the best BLEU score. This implies that good monolingual translation is not necessarily good simplification. OOV is the percentage of words that are not in the Basic English BE850 list.[10] TSM is ranked as the second best system for this criterion.

The perplexity (PPL) is a score of text probability measured by a language model and normalized by the number of words in the text (Equ. 6). PPL can be used to measure how tight the language model fits the text. Language models constitute an important feature for assessing readability (Schwarm and Ostendorf, 2005). We train a trigram LM using the simple sentences in PWKP and calculate the PPL with SRILM. TSM gets the best PPL score. From this table, we can conclude that TSM achieves better overall readability than the baseline systems.

$$PPL(text) = P(w_1 w_2 ... w_N)^{-\frac{1}{N}} \quad (6)$$

There are still some important issues to be considered in future. Based on our observations, the current model performs well for word substitution and segmentation. But the completion of the new sentences is still problematic. For example, we copy the dependent NP to the new sentences. This may break the coherence between sentences. A better solution would be to use a pronoun to replace the NP. Sometimes, excessive droppings occur, e.g., "older" and "twin" are dropped in "She has an older brother and a twin brother...". This results in a problematic sentence: "She has an brother and a brother...". There are also some errors which stem from the dependency parser. In Example 2, "An umbrella term" should be a dependency of "called". But the parser returns "superset" as the dependency. In the future, we will investigate more sophisticated features and rules to enhance TSM.

7 Conclusions

In this paper, we presented a novel large-scale parallel dataset PWKP for sentence simplification. We proposed TSM, a tree-based translation model for sentence simplification which covers splitting, dropping, reordering and word/phrase substitution integrally for the first time. We also described an efficient training method with speeding up techniques for TSM. The evaluation shows that TSM can achieve better overall readability scores than a set of baseline systems.

[10] http://simple.wikipedia.org/wiki/Wikipedia:Basic_English_alphabetical_wordlist

References

Barzilay, Regina and Noemie Elhadad. 2003. Sentence alignment for monolingual comparable corpora. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 25–32.

Carroll, John, Guido Minnen, Darren Pearce, Yvonne Canning, Siobhan Devlin, and John Tait. 1999. Simplifying text for language-impaired readers. In *Proceedings of the 9th Conference of the European Chapter of the Association for Computational Linguistics (EACL'99)*, pages 269–270.

Chandrasekar, R., Christine Doran, and B. Srinivas. 1996. Motivations and methods for text simplification. In *Proceedings of the Sixteenth International Conference on Computational Linguistics (COLING'96)*, pages 1041–1044.

Filippova, Katja and Michael Strube. 2008a. Dependency tree based sentence compression. In *International Natural Language Generation Conference (INLG'08)*, pages 25–32.

Filippova, Katja and Michael Strube. 2008b. Sentence fusion via dependency graph compression. In *EMNLP '08: Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 177–185.

Graehl, Jonathan, Kevin Knight, and Jonathan May. 2008. Training tree transducers. In *Computational Linguistics*, volume 34, pages 391–427. MIT Press.

Heilman, M. and N. A. Smith. 2009. Question generation via overgenerating transformations and ranking. Technical Report CMU-LTI-09-013, Language Technologies Institute, Carnegie Mellon University.

Inui, Kentaro, Atsushi Fujita, Tetsuro Takahashi, Ryu Iida, and Tomoya Iwakura. 2003. Text simplification for reading assistance: A project note. In *Proceedings of the 2nd International Workshop on Paraphrasing: Paraphrase Acquisition and Applications (IWP)*, pages 9–16.

Jonnalagadda, Siddhartha and Graciela Gonzalez. 2009. Sentence simplification aids protein-protein interaction extraction. In *Proceedings of the 3rd International Symposium on Languages in Biology and Medicine*.

Jurafsky, Daniel and James H. Martin. 2008. *Speech and Language Processing (2nd Edition)*. Prentice Hall, 2 edition.

Klein, Dan and Christopher D. Manning. 2003. Fast exact inference with a factored model for natural language parsing. In *Advances in Neural Information Processing Systems 15 (NISP'02)*, pages 3–10.

Knight, Kevin and Daniel Marcu. 2000. Statistics-based summarization - step one: Sentence compression. In *AAAI*, pages 703–710.

Levenshtein. 1966. Binary code capable of correcting deletions, insertions and reversals. In *Soviet Physics*, pages 707–710.

Nelken, Rani and Stuart M. Shieber. 2006. Towards robust context-sensitive sentence alignment for monolingual corpora. In *Proceedings of 11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 161–168.

Petersen, Sarah E. and Mari Ostendorf. 2007. Text simplification for language learners: a corpus analysis. In *Proc. of Workshop on Speech and Language Technology for Education*, pages 69–72.

Schmid, Helmut. 1994. Probabilistic part-of-speech tagging using decision trees. In *International Conference on New Methods in Language Processing*, pages 44–49.

Schwarm, Sarah E. and Mari Ostendorf. 2005. Reading level assessment using support vector machines and statistical language models. In *ACL'05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 523–530.

Siddharthan, Advaith. 2002. An architecture for a text simplification system. In *Proceedings of the Language Engineering Conference (LEC'02)*, pages 64–71.

Siddharthan, Advaith. 2006. Syntactic simplification and text cohesion. In *Research on Language & Computation*, volume 4, pages 77–109. Springer Netherlands, June.

Stolcke, Andreas. 2002. SRILM - An Extensible Language Modeling Toolkit. pages 901–904.

Vickrey, David and Daphne Koller. 2008. Sentence simplification for semantic role labeling. In *Proceedings of ACL-08: HLT*, pages 344–352, June.

Watanabe, Willian Massami, Arnaldo Candido Junior, Vinícius Rodriguez Uzêda, Renata Pontin de Mattos Fortes, Thiago Alexandre Salgueiro Pardo, and Sandra Maria Aluísio. 2009. Facilita: reading assistance for low-literacy readers. In *SIGDOC '09: Proceedings of the 27th ACM international conference on Design of communication*, pages 29–36. ACM.

Yamada, Kenji and Kevin Knight. 2001. A syntax-based statistical translation model. In *ACL'01: Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 523–530.

Yamada, Kenji and Kevin Knight. 2002. A decoder for syntax-based statistical mt. In *ACL'02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 303–310.

Zesch, Torsten, Christof Müller, and Iryna Gurevych. 2008. Extracting Lexical Semantic Knowledge from Wikipedia and Wiktionary. In *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, pages 1646–1652.

Zhao, Shiqi, Xiang Lan, Ting Liu, and Sheng Li. 2009. Application-driven statistical paraphrase generation. In *Proceedings of ACL-IJCNLP*, pages 834–842, Suntec, Singapore, August.