DEPSYM: A Lightweight Syntactic Text Simplification Approach using Dependency Trees

Niladri Chatterjee, Raksha Agarwal

Indian Institute of Technology Delhi, Hauz Khas, Delhi-110016, India

Abstract

Syntactic Simplification typically involves sentence splitting, changing of voice from passive to active, and resolving other ambiguities. The present work proposes a lightweight syntactic simplification algorithm, named DEPSYM, which uses the dependency parse tree of a complex sentence for simplification and splitting. Automatic and Human evaluations on two commonly used datasets indicate that the proposed system produces structurally simpler outputs while preserving grammaticality and meaning. The proposed system is designed for simplification of English sentences.

Keywords

Dependency Tree, Syntactic Simplification, Sentence Segmentation

1. Introduction

The task of Text Simplification (TS) aims at modifying textual content in order to improve its readability while preserving the meaning. Text Simplification can be categorized as *Lexical Simplification*, *Syntactic Simplification*, *Explanation Generation*, and *Monolingual Machine Translation* [1]. Lexical Simplification aims at identifying and replacing complex words (or phrases) by an easier-to-read alternative. The alternative can be a single word or a phrase. Syntactic Simplification (SS) simplifies the text by restructuring the words of the sentences, and/or rewriting it into smaller sentences. Explanation Generation systems take difficult concepts of a text and augment it with extra information in order to improve the context and user understanding, instead of just rewriting the input text. Monolingual Machine Translation based approaches treat the simplification problem as a translation problem from complex language to simpler language. This method usually simplifies both the lexicon and syntax of a text.

Long and syntactically complex sentences are difficult to understand for second language learners [2] and people with reading difficulties [3, 4]. Syntactic Simplification typically involves sentence splitting, changing passive voice to active and resolving other ambiguities. The aim of the present work is to perform syntactic simplification of English sentences using their respective dependency tree structure. The proposed system focuses on rewriting along with splitting of sentences, if required, which contain *passive voice*, *appositive*, *relative* and *conjoint* clauses. It is considered lightweight in the sense that it relies only on the dependency structure of the sentence for syntactical transformations. It does not adopt more intensive simplification

Proceedings of the First Workshop on Current Trends in Text Simplification (CTTS 2021), co-located with SEPLN 2021. September 21st, 2021 (Online). Saggion, H., Štajner, S. and Ferrés, D. (Eds).

niladri.chatterjee@maths.iitd.ac.in (N. Chatterjee); raksha.agarwal@maths.iitd.ac.in (R. Agarwal)

strategies, such as event-deletion, anaphora resolution, rhetorical structure theory that are being carried out in many other syntactic simplification systems as discussed in Section 2.

The rest of the paper is organised as follows. Existing related works are briefly described in Section 2. Section 3 presents the proposed approach. Experimental details are described in Section 4, and Section 5 presents the results. The paper is concluded in Section 6.

2. Related Work

The earliest work on Syntactic Simplifcation by Chandrasekar et al. [5] developed rules for transformation of sentences containing relative clauses or appositions. It used Finite State Grammar to produce noun and verb groups, and a Supertagging model to produce dependency linkages via partial parsing. Chandrasekar et al. [6] automated the rule generation system for relative clauses using parallel data of 65 sentences. The transformation rules are identified using a tree-comparison algorithm, and they are generalized by replacing specific words by tags.

PSET developed by Caroll et al. [7] is the first assistive simplification system aimed for the benefit of people with aphasia. PSET applies both syntactic and lexical simplification on English newspaper texts as available on the Internet. Here, the sentences are converted into subject-verb-object order by replacing passive constructions with active ones. This was further improved using SYSTAR [8] to split compound sentences. It also converted seven passive clause types to active, and resolved and replaced eight frequently occurring anaphoric pronouns.

Previous research on text simplification did not account for the discourse level issues related of conjunction and anaphora that arise from applying syntactic transforms at the sentence level. In order to improve the cohesion in the simplified text, Siddharthan introduced a regeneration step in addition to transformation step [9, 10]. It preserves conjunctive cohesion by using rhetorical structure theory for cue-word selection, sentence ordering and choice of determiner. Glavas et al. [11] performed simplification of sentences using an event-centered approach. Their system extracted events from a given sentence; and discarded text that did not belong to any of the factual events. However, it is an event-based simplification and conceptually different from lexical or syntactic simplification.

REGENT [12] applied transformation rules for syntactic simplification to typed dependency representations produced by the Stanford parser [13]. Simplification is performed for sentences containing coordination, subordination, apposition, passive constructions, and relative clauses. An overgenerate-and-rank approach was adopted by performing simplification on top-n parses of a sentence in order to reduce errors due to inaccurate parsing. The outputs were evaluated by determining the extent of simplification and level of accuracy in rule application.

Siddharthan et al. [14] developed a hybrid text simplification system using synchronous dependency grammars with hand-written and automatically harvested rules. This allowed for a linguistically sound treatment of complex sentences requiring reordering and morphological changes, such as conversion of passive voice to active.

Ferrés et al. [15] developed a hybrid text simplifier for English. Their syntactic simplification system used rule-based analysis and generation techniques based on PoS tags and dependency trees. Intrinsic evaluation on a self-collected dataset of 500 sentences suggested that correct simplifications were obtained for 74.2% sentences. In this system 66 rules have been used

Table 1
Examples

	Complex	Simple
Appositive	Bob, 61 years old, will join the company.	Bob will join the company. Bob is 61 years old.
Relative	Bob, who I live with, is a very nice man.	Bob is a very nice man. I live with Bob.
Conjoint	Bob does not have a cat but, has a parrot.	Bob does not have a cat. Bob has a parrot.
Passive	The letter is being written by Bob.	Bob is writing the letter.

to simplify seven syntactic phenomena, namely appositive, relative, sentence coordination, coordinated correlatives, subordination, adverbial, and passive voice.

Scarton et al. [16] developed a multilingual syntactic simplification tool (MUSST) using dependecy based rules. The rules simplify sentences containing conjoint clauses, relative clauses, appositive phrases and/or passive voice. The order of clauses were not taken into consideration when complex marker, such as *although* is replaced by a simpler marker *but*. An accuracy of 76% was reported for simplifying a set of 292 English sentences. However, system evaluations of this work for standard datasets, such as PWKP [17] and TurkCorpus [18] are not available.

Syntactic Simplification systems have also been developed for non-English languages, such as Spanish [19], Italian [20], Portuguese [21]. The SIMPLEXT system [19] utilizes the dependency structure of Spanish sentences to perform five types of syntactic transformation, such as separation of participial modifiers, splitting relative and coordinating clauses, reordering of quoted objects. ERNESTA [20] performs simplification of Italian sentences using an event-based approach and it focuses on anaphora resolution and syntactic simplification while removing unnecessary events. PORSIMPLES [21] is a rule-based simplification system for Brazilian Portuguese for simplifications of appositive, relative, coordinate and subordinate clauses. Additionally, sentence splitting and reordering of clauses were also performed.

More recently, the focus of researchers has shifted to Hybrid and Neural Text Simplification systems which perform both lexical and syntactic simplifications [22, 23, 24, 25, 26, 27]. In the next section the proposed syntactic simplification algorithm, DEPSYM (DEpendency Parse based SYntactic siMplification) is discussed.

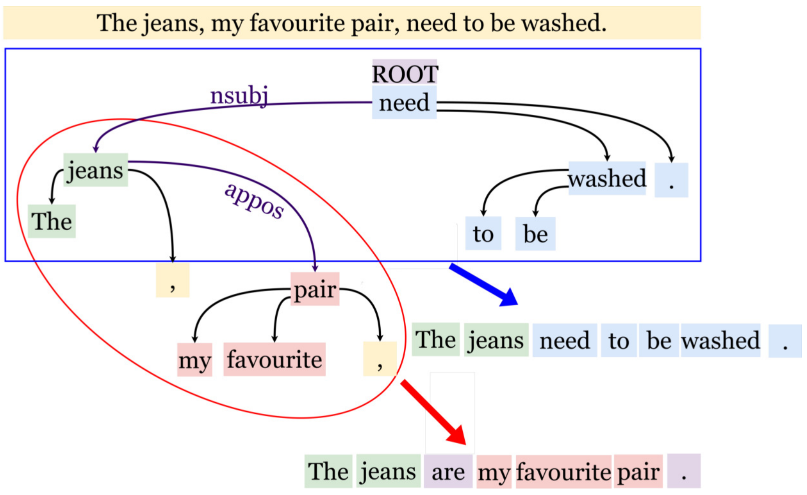
3. The Proposed Approach: DEPSYM

In the present work, rules based on dependency trees extracted from spaCy[1] are utilized for simplification of appositive clauses, relative clauses, conjoint clauses, and passive-to-active conversion (See Table 1). Other dependency parsers for English (e.g. Stanford) may also be utilised for extraction of dependency trees but, since the proposed system is developed using Python, spaCy (version:2.2.4, en_core_web_sm) has been used for ease of implementation. It uses a non-monotonic arc-eager transition-system [28] for parsing sentences. Below we explain the different modules of the proposed approach.

[1]https://github.com/explosion/spaCy

3.1. Appositive Clauses

In the first step, the appositive token, the one with *appos* tag, is extracted from the dependency tree of the sentence. If such a token is not found then token with *amod* tag is extracted. If neither of the two tags are found then appositive clause is not present in the sentence. The parent of the appositive token may be subject (with *nsubj* tag) or the object (with *dobj*/*pobj* tag). An in-order traversal in the sub-tree of the appositive token gives the appositive phrase. The noun phrase is determined by taking the left sub-tree traversal of the parent of the appositive token. The required auxiliary verb is determined using the tense of the root, and the singularity/plurality of the main subject. The appositive token and its sub-tree are deleted from the main parse tree. The first sentence is extracted by in-order traversal of the main tree. The second sentence is formed by combining the noun phrase, auxiliary verb, and the appositive phrase (See Figure 1).



The jeans, my favourite pair, need to be washed.
ROOT
need
nsubj
jeans
The
appos
washed
.
to
be
,
pair
my
favourite
,
The jeans need to be washed .
The jeans are my favourite pair .

Figure 1: Simplification of sentence containing appositive clause

3.2. Relative Clauses

The relative token, the one with *relcl* tag, is extracted from the dependency tree of the sentence. If such a token is not found then token with *rcmod* tag is extracted. If neither of the two tags are found then it is concluded that relative clause is not present in the sentence. The parent of the relative token gives the subject/object that the relative clause refers to. The left sub-tree of the parent gives the noun phrase. Inorder traversal of the relative token's sub-tree gives the relative clause. The relative token is then deleted from the main tree, and subsequent inorder traversal of the main tree gives the first sentence. The second sentence is formed by combining the noun phrase and the relative phrase. The noun phrase is not inserted within the relative phrase but the relative pronoun is retained in the relative phrase. The order of the two sentences is decided on the basis of whether the relative clause is attached to the subject or to the object (See Figure 2).

For illustration, the sentence *Bob, who I live with in Delhi, is a nice man.* is simplified as *Bob* *who I live with in Delhi. Bob is a nice man.*, and the sentence *Bob lives in Delhi, where his brother lives.* is simplified as *Bob lives in Delhi. Delhi is where his brother lives.*

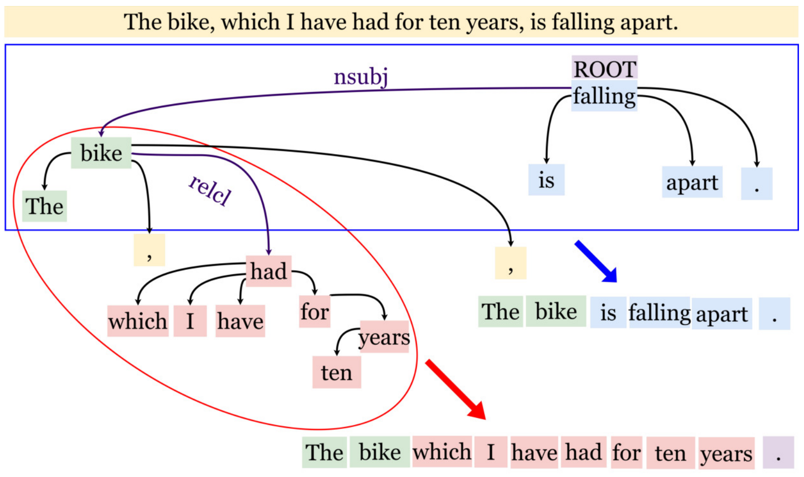

The bike, which I have had for ten years, is falling apart.
nsubj
ROOT
falling
bike
The
relcl
is
apart
.
,
had
which
I
have
for
years
ten
,
The bike is falling apart .
The bike which I have had for ten years .

Figure 2: Simplification of sentence containing relative clause

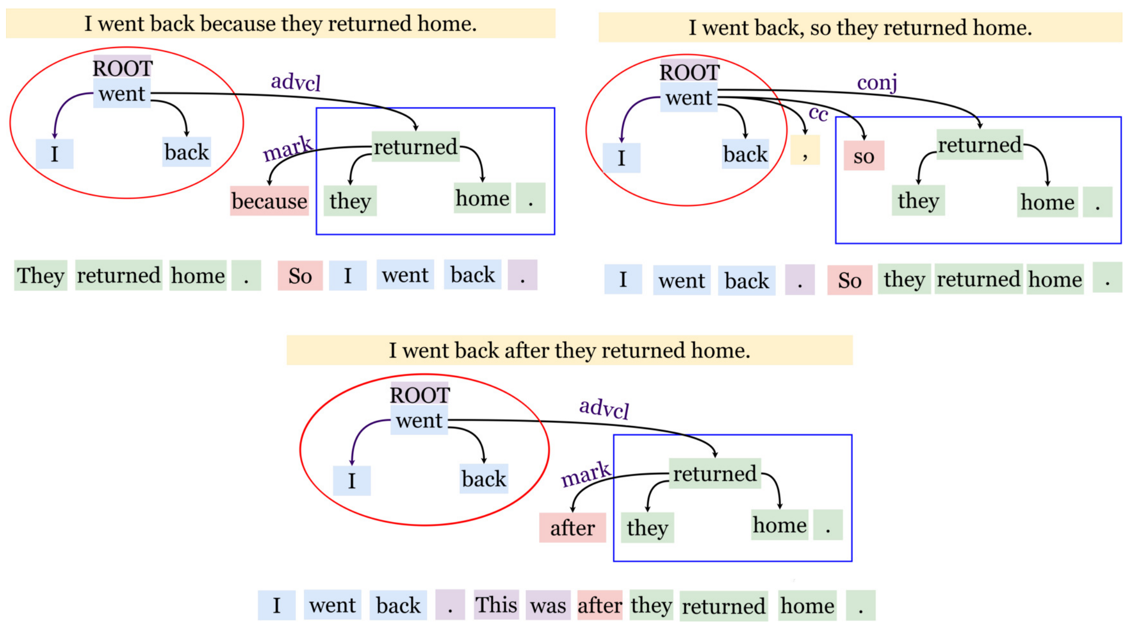
3.3. Conjoint Clauses

Conjoint clauses may contain a variety of conjunctions, such as *though*, *although*, *when*, *because*. A sentence with a conjoint structure is typically made of two clauses[2] connected by a conjunction or an adverbial modifier. The dependency tags, namely *conj*, *advcl*, *parataxis*, and *ccomp*, connect the root of the sentence to the root of the second clause. The conjunction term or adverbial modifier is connected to the root of the second clause with a *cc* or *advmod* or *mark* tag. This token and root of the conjoint clause is removed from the main tree. If the conjoint clause has no subject then it is given the subject of the main clause (sentence). Inorder traversal of the main tree and the sub-tree of conjoint clause root gives the two separate sentences. The order of the two sentences depends on the type of conjunction and the position of the conjunction in the sentence. Conjunctions, such as *although*, *whereas*, *however* are replaced with *but*. Conjunctions *because* and *as* are replaced with the conjunction *so*. Further, the order of the two related sentences (clauses) are reversed. For conjunctions *before*, *after*, *once*, *since* and *when*, the phrase *this + auxiliary verb* is inserted in the second sentence as shown in Figure 3.

3.4. Passive Voice

Passive to active voice algorithm contains three modules, namely *subject-object detection*, *verb tense transform*, and *pronoun form transform*. To identify a passive sentence, the algorithm searches for a token with *auxpass* tag in the dependency tree. This token is a verb, but not the main verb, of the clause which contains the passive information. This is followed by a search for tokens with tag *agent* and *nsubjpass*. Absence of any of these results in no simplification. All

[2]as observed in the development phase



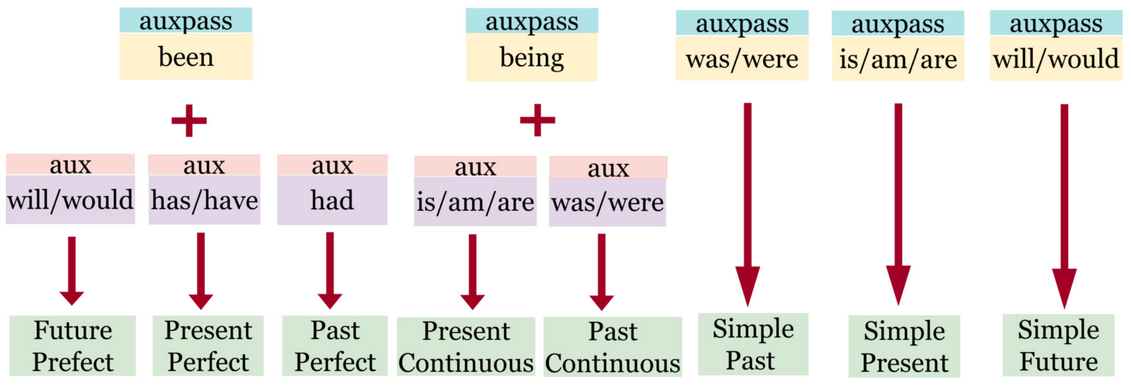
I went back because they returned home.
ROOT
went
advcl
I
back
mark
returned
because
they
home
.
They returned home . So I went back .
I went back, so they returned home.
ROOT
went
conj
cc
I
back
,
so
returned
they
home
.
I went back . So they returned home .
I went back after they returned home.
ROOT
went
advcl
I
back
mark
returned
after
they
home
.
I went back . This was after they returned home .

Figure 3: Simplification of sentences containing conjoint clause

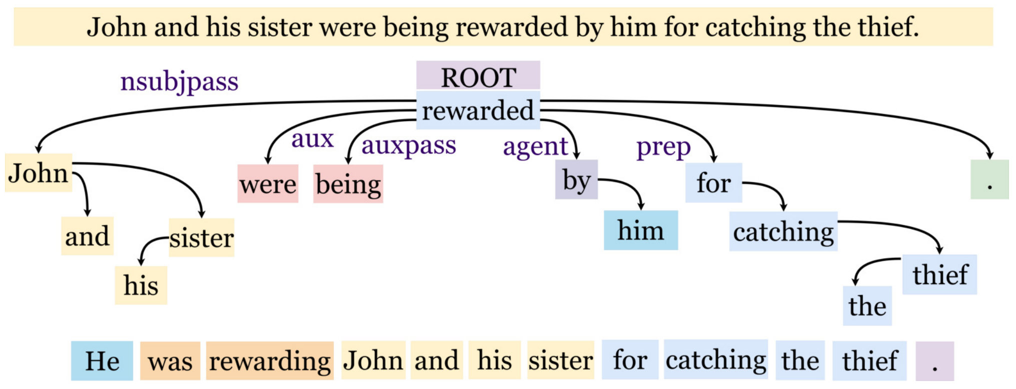
the auxiliary verbs (token with *aux* tag) of the sentence and *auxpass* token are removed from the tree. Inorder traversal of *nsubjpass* tag token determines the object phrase for the active voice sentence. The subject phrase for the active voice sentence is obtained by first finding *pobj* dependency in *agent* token's children and then carrying out an inorder-traversal of its subtree. After that, *agent* and *nsubjpass* tokens are removed from the main tree.

The second module takes as input five parameters, namely the main verb (*root*), auxiliary verbs, *auxpass* token, subject token and subject POS tag. These are used to determine the tense of the sentence as described in Figure 4a, and also helps to decide the new form for the verb and the auxiliary verbs. Transformation of verb forms due to presence of pronouns and modal verbs are also performed. The third module is used to change the form of pronouns that are moving from subject to object or vice-versa. A dictionary containing common pronoun pairs, such as *he: him*, *she: her*, *we: us*, is used to carry out the transformation. Finally, the new sentence is constructed by interchanging the positions of subject-object in the tree and inserting the new verb. Other dependencies of the root are maintained in their positions; but any prepositional phrases to the right of root are moved to the end (See Figure 4b). Simplification of a complex sentence is performed recursively in the order appositive phrases, conjoint clauses, relative clauses and then passive voice. Finally, true casing[3] is performed.

[3]using https://pypi.org/project/truecase/



(a) Rules for Sentence Tense Identification


John and his sister were being rewarded by him for catching the thief.
nsubjpass
ROOT
rewarded
aux
auxpass
agent
prep
John
and
sister
his
were
being
by
him
for
catching
thief
the
.
He was rewarding John and his sister for catching the thief .

(b) Passive to Active conversion

Figure 4: Simplification of Passive voice sentences

4. Experimental Details

This section discusses experimental details about the datasets, baseline systems, automatic evaluation metrics and human evaluation procedure.

4.1. Dataset

The proposed simplification algorithm has been evaluated on two commonly used test sets[4] extracted from English Wikipedia (EW) and Simple English Wikipedia (SEW) namely, PWKP [17] and TurkCorpus [18].

- **PWKP/WikiSmall:** The test set for this dataset contains 100 sentences with only one simplification reference per original sentence. The references are extracted by aligning sentences of EW and SEW.
- **TurkCorpus (TC):** The test set for this dataset contains 359 complex sentences. The following reference sets have been considered for this dataset:

[4]Retrieved from https://github.com/feralvam/easse/tree/master/easse/resources/data/test_sets

- In **MTurk** [18], eight mannually written references focused on Lexical simplification are collected using Amazon Mechanical Turk for each sentence of TC . Here, the annotators were instructed to rewrite sentences by reducing the number of difficult words or idioms, but without deleting content or splitting the sentences.
- **HSplit** [29] provides four reference simplifications for TC specifically for assessing sentence splitting. Two annotators were instructed to perform two tasks: (1) split the original sentence as much as possible, while preserving grammaticality, fluency and meaning, (2) split the sentence only when it simplifies the original sentence.
- **ASSET** [30] contains ten references for TC focusing on several rewriting transformations, e.g. lexical simplification and reordering, sentence splitting, compression.

4.2. Baselines

The following baselines[5] have been considered in the present work.

- **Identity:** This refers to the system where output is identical to the input.
- **TSM:** It aims at finding the best sequence of transformation of the constituency parse tree of the complex sentence to produce the target simplification [17]. It is trained on the train subset of PWKP.
- **PBSMT-R:** It uses Phrase Based Statistical Machine translation along with a dissimilarity-based (measured using edit distance) reranking mechanism to chose among possible simplifications [31].
- **HYBRID:** It combines deep semantics and monolingual machine translation to derive simple sentences from complex sentences [23]. Semantic role information is used to determine events in the sentence for performing splitting and deletion. For substitution of complex words and sentence reordering a PBSMT-based model is used.
- **DSS:** It is a sentence splitting algorithm which uses the UCCA semantic parser [32] to decompose a sentence to its main semantic constituents [26].
- **YATS:** It is a hybrid text simplification system which also uses a rule-based syntactic simplification system along with lexical simplification [15].
- **DRESS:** It uses a Reinforcement Learning (RL) architecture with standard attention-based encoder-decoder as an agent. The reward function uses a weighted sum of SARI [18], cosine similarity of input and output, and language model probability of output [24].
- **ACCESS:** It uses a discrete parametrization mechanism to improve performance of Seq2Seq models for the task of simplification. It adds control tokens corresponding to four attributes, namely amount of compression, amount of paraphrasing, lexical complexity and syntactical complexity [27].

4.3. Evaluation

For automatic evaluation of system outputs commonly used metrics, namely FKGL, BLEU and SARI have been utilized.

[5]Baseline system outputs obtained from https://github.com/eliorsulem/simplification-acl2018/tree/master/All_system_outputs for DSS, http://able2include.taln.upf.edu/ for YATS, and https://github.com/feralvam/easse/tree/master/easse/resources/data/system_outputs for the rest

Table 2
Questions for Human Evaluation

Fluency (Fl)	Is the output grammatical and well formed?
Adequacy (Aq)	Does the simplification preserve the meaning of the original sentence?
Simplicity (Sp)	Is the simplification easier to understand than the input?
Structural Simplicity (StS)	Is the output simpler than the input, ignoring the complexity of the words?

- **F**lesch-**K**incaid **G**rade **L**evel (**FKGL**) [33]: This metric corresponds to the grade level of a sentence. Lower FKGL implies simpler outputs and lower level of difficulty.
- **B**i**L**ingual **E**valuation **U**nderstudy (**BLEU**) [34]: This is a precision-oriented metric used to measure the correctness of the generations by measuring n-grams overlaps between the generated sentences and (multiple) references.
- **S**ystem output **A**gainst **R**eferences and **I**nput sentence (**SARI**) [18]: This metric compares model-generated simplifications with both the input sentence and the gold references. It measures “how good” the words added, deleted, and kept by a simplification model are.

The above-mentioned metrics have been calculated using EASSE implementation [35]. Additionally, ten human annotators were asked to evaluate the system outputs on a 5-point Likert scale on the basis of Fluency, Adequacy, Simplicity, and Structural Simplicity as described in Table 2. Structural Simplicity is distinguished from Simplicity in order to ignore the effect of lexical simplification performed by the baseline systems. The annotators were non-native but fluent English speakers. Each annotator scored 260 system outputs[6] correponding to 20 input sentences of each test dataset. YATS system was evaluated by two of the above annotators[7]. Fluency and Adequacy were measured using a 1 to 5 scale. Following previous works [26, 25], a five point scale [-2, 2] is used to measure Simplicity and Structural Simplicity, where a 0 score indicates that the input and the output are equally complex. A score of 1 (-1) indicates that the output is slightly simple (complex) than input, and score of 2 (-2) indicates that the output is very simple (complex) than input. The advantage of using a -2 to 2 scale is that the sign of the score could indicate the efficiency of the simplification system. Negative scores indicate that the output is more complex than the input sentence.

5. Results and Analysis

The results for PWKP test set and TC test set are presented in Table 3 and Table 4, respectively. The DEPSYM system[8] proposed in this work performs well in terms of FKGL for both the test sets. The FKGL is reduced nearly by 50 percent with respect to the input sentences (Identity) for both the test sets for DEPSYM. However, since DEPSYM does not perform lexical simplifications or phrase deletions, BLEU and SARI scores are lower for PWKP, MTurk, and ASSET data. For TurkCorpus, BLEU scores for the DEPSYM system are consistent throughout the three reference sets. Moreover, it is higher than DSS and HYBRID systems. The BLEU and SARI scores obtained

[6]PWKP: 120 (6×20) +TurkCorpus: 140 (7×20)
[7]as on 2 September 2021
[8]Code and System Outputs are available at https://github.com/RakshaAg/DEPSYM

Table 3
Results for PWKP/WikiSmall test set

System	FKGL	BLEU	SARI	Fl	Aq	Sp	StS
Identity	13.10	49.07	22.27	4.92±0.10	5±0.00	0±0.00	0±0.00
PBSMT-R	12.26	**45.49**	35.49	4.65±0.18	**4.55**±0.19	0.31±0.12	0.05±0.10
HYBRID	10.29	44.20	**54.67**	3.28±0.60	3.47±0.53	-0.05±0.29	-0.19±0.38
YATS	9.12	37.99	35.87	3.88±0.18	3.78±0.18	-0.20±0.14	**0.78**±0.39
DRESS	7.48	34.53	27.48	**4.83**±0.20	3.15±0.58	**0.59**±0.35	0.52±0.38
TSM	**6.40**	30.75	39.02	3.46±0.53	3.19±0.57	-0.14±0.30	0.03±0.44
DEPSYM	6.67	36.84	33.78	4.14±0.51	**4.55**±0.46	0.27±0.49	0.54±0.65

by DEPSYM is competitive with the YATS system. DEPSYM significantly increases the SARI score with respect to the input sentences for both the test sets. For HSplit, the proposed DEPSYM system achieves the second highest BLEU and SARI score among all the baselines.

Out of the 459 sentences belonging to the two test datasets, PWKP and TC, DEPSYM performed simplification of 295 sentences, 79% of which were correctly simplified. Out of 100 sentences of PWKP test set, 80 were simplified. The simplifications were correct for 57 input sentences. For the TC test set out of 215 simplifications, 176 sentences were correct.

Human Evaluation indicates that in terms of Adequacy, the proposed system outperforms other baseline systems. With respect to Structural Simplicity DEPSYM performs the best for TurkCorpus, and for PWKP it came second highest. For Fluency and Simplicity, the values achieved by DEPSYM is highly competitive beating most of the baseline systems comfortably.

ACCESS and DRESS systems perform lexical substitution of the complex words, and thereby achieved higher scores with respect to simplicity. The outputs of DRESS are also rated as most fluent by the annotators. However, these systems perform poorly with respect to Adequacy as they often delete important phrases from the original sentence. Moreover, sometimes the lexical changes performed by these system lead to semantic errors in the output. For illustration, consider the simplified outputs for DEPSYM and baseline systems in Table 5. Here, HYBRID system omits important information, while other baseline systems replaces the word *distributes* incorrectly by *collects* or *deals* or *sells.* It can also be observed that the output obtained using the neural ACCESS system contain incorrect repetition of the phrase *present location.* The output of DSS system, which is developed for sentence splitting, contains an incomplete sentence '*was added in 1938-39*'. The YATS system failed to simplify the conjoint clause present in the second example. In both the examples mentioned in Table 5, DEPSYM produces structurally simpler outputs with two or more smaller sentences.

6. Conclusion

Simplification of long and syntactically complex sentences is an important NLP task in order to facilitate accessibility to a larger set of audience. In the present work, syntactic simplification is performed by analyzing the dependency tree structure of a complex English sentence. The relationship between words in the dependency tree is utilised to split sentences containing any of

Table 4
Results for TurkCorpus test set

		MTurk		ASSET		HSplit					
System	FKGL	BLEU	SARI	BLEU	SARI	BLEU	SARI	Fl	Aq	Sp	StS
Identity	10.02	99.36	26.29	92.56	20.73	81.53	30.11	4.93±0.08	5±0.00	0±0.00	0±0.00
PBSMT-R	8.85	**81.81**	38.04	78.56	34.63	62.42	35.24	4.30±0.34	4.38±0.33	0.42±0.19	0.06±0.21
YATS	7.65	46.99	38.43	70.51	36.72	**65.57**	44.48	4.33±0.18	4.43±0.25	-0.03±0.11	0.70±0.07
DRESS	7.53	77.47	36.84	**83.32**	37.07	48.69	37.21	**4.83**±0.11	3.75±0.34	0.45±0.26	0.35±0.25
ACCESS	7.29	75.77	**41.38**	75.39	**40.13**	57.97	38.22	4.49±0.37	3.91±0.49	**0.49**±0.33	0.19±0.15
DSS	6.22	59.66	36.30	56.68	35.24	58.51	**48.59**	3.16±0.80	4.20±0.49	-0.17±0.40	-0.05±0.38
HYBRID	**5.18**	49.76	31.50	56.07	34.65	24.70	25.90	3.01±0.57	2.57±0.58	-0.45±0.44	-0.29±0.48
DEPSYM	5.69	68.64	34.68	65.15	31.07	62.85	45.27	4.09±0.46	**4.61**±0.36	0.43±0.41	**0.82**±0.49

appositive, relative or conjoint clauses. Furthermore, sentence rewriting from passive to active voice is also performed. Automatic and Human evaluations indicate that the proposed system produces structurally simpler outputs while preserving grammaticality and meaning. The proposed simplification system may be applicable to other languages having similar dependency structure as English. For languages where the dependency structures are different the rules will have to be fine-tuned in keeping with the linguistic properties of the language concerned.

The proposed DEPSYM algorithm uses the dependency parsing of the input sentence. Hence, parsing errors induced by SpaCy lead to some errors in simplification albeit inadvertently. In future, the passive to active voice conversion may be further improved to handle negative sentences. For illustration, *He was not chased by the the police* is incorrectly simplified as *The police not chased him* as it is unable to perform the transformation *was not chased* → *did not chase*. For sentences with relative clauses, the relative pronouns are retained in the simplified sentence. This can be improved by curating rules for insertion of the noun phrase within the relative phrase. For illustration, *Bob, who I live with in Delhi, is a nice man.* may be simplified as *I live with Bob in Delhi. Bob is a nice man.* In future we would also like to append a lexical simplification module with the current syntactic simplification algorithm.

Acknowledgments

The authors thank Daanish Bansal and Siddhant for helpful discussions. Raksha Agarwal acknowledges Council of Scientific and Industrial Research (CSIR), India for supporting the research under Grant no: SPM-06/086(0267)/2018-EMR-I. We thank Prof. Horacio Saggion for providing a working demo of YATS system. We also thank the reviewers for their valuable suggestions and constructive criticism.

References

[1] M. Shardlow, A survey of automated text simplification, International Journal of Advanced Computer Science and Applications 4 (2014) 58–70.

Table 5
System Outputs

System	Output
Identity (PWKP)	This array distributes data across multiple disks, but the array is seen by the computer user and operating system as one single disk.
PBSMT-R	This array distributes data across multiple disks, but the array is seen by the computer user and operating system as a single disk.
HYBRID	this array releases data across disks, but the array is seen by the user and computer operating system.
DRESS	This array *collects* data across multiple disks, but the array is seen by the computer user and operating system.
TSM	This array *sells* data across multiple disks but the array is seen. The computer user and operating as one disk.
YATS	This array *deals* data across multiple disks, but the computer user and operating system sees the array as one single disk.
DEPSYM	This array distributes data across multiple disks. **But, the computer user and operating system sees the array as one single disk.**
Identity (TC)	The fourth ring is decorated with golden garlands and was added in 1938-39 when the column was moved to its present location.
PBSMT-R	The fourth ring is decorated with golden garlands was added in 1938-39 when the column was moved to its present location.
DRESS	The fourth ring is decorated with golden garlands.
ACCESS	The fourth ring was added in 1938 when the column was moved to its present location, and was added to its present location.
DSS	the fourth ring is decorated with golden garlands. was added in 1938-39. the column was moved to its present location.
HYBRID	The ring is decorated and was added the column was moved.
YATS	The fourth ring is decorated with golden Garlands. And the fourth ring was added in 1938 - 39 when the column was moved to its present location.
DEPSYM	The fourth ring is decorated with golden garlands. And ring was added in 1938-39. **This was when** the column was moved to its present location.

[2] H. Saggion, Automatic text simplification, Synthesis Lectures on Human Language Technologies 10 (2017) 1–137.

[3] S. Devlin, G. Unthank, Helping aphasic people process online information, in: Proceedings of the 8th International ACM SIGACCESS Conference on Computers and Accessibility, 2006, pp. 225–226.

[4] V. Yaneva, I. Temnikova, R. Mitkov, Evaluating the readability of text simplification output for readers with cognitive disabilities, in: Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16), 2016, pp. 293–299.

[5] R. Chandrasekar, C. Doran, S. Bangalore, Motivations and methods for text simplification, in: COLING 1996 Volume 2: The 16th International Conference on Computational Linguistics, 1996.

[6] R. Chandrasekar, B. Srinivas, Automatic induction of rules for text simplification,

Knowledge-Based Systems 10 (1997) 183–190.

[7] J. Carroll, G. Minnen, Y. Canning, S. Devlin, J. Tait, Practical simplification of english newspaper text to assist aphasic readers, in: Proceedings of the AAAI-98 Workshop on Integrating Artificial Intelligence and Assistive Technology, 1998, pp. 7–10.

[8] Y. Canning, J. Tait, J. Archibald, R. Crawley, Cohesive generation of syntactically simplified newspaper text, in: International Workshop on Text, Speech and Dialogue, Springer, 2000, pp. 145–150.

[9] A. Siddharthan, Syntactic simplification and text cohesion, Research on Language and Computation 4 (2006) 77–109.

[10] A. Siddharthan, An architecture for a text simplification system, in: Language Engineering Conference, 2002. Proceedings, 2002, pp. 64–71. doi:10.1109/LEC.2002.1182292.

[11] G. Glavaš, S. Štajner, Event-centered simplification of news stories, in: Proceedings of the Student Research Workshop associated with RANLP 2013, 2013, pp. 71–78.

[12] A. Siddharthan, Text simplification using typed dependencies: A comparision of the robustness of different generation strategies, in: Proceedings of the 13th European Workshop on Natural Language Generation, 2011, pp. 2–11.

[13] M.-C. De Marneffe, B. MacCartney, C. D. Manning, Generating typed dependency parses from phrase structure parses, in: Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06), European Language Resources Association (ELRA), Genoa, Italy, 2006. URL: http://www.lrec-conf.org/proceedings/lrec2006/pdf/440_pdf.pdf.

[14] A. Siddharthan, A. Mandya, Hybrid text simplification using synchronous dependency grammars with hand-written and automatically harvested rules, in: Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, 2014, pp. 722–731.

[15] D. Ferrés, M. Marimon, H. Saggion, A. AbuRa'ed, Yats: yet another text simplifier, in: International Conference on Applications of Natural Language to Information Systems, Springer, 2016, pp. 335–342.

[16] C. Scarton, A. P. Aprosio, S. Tonelli, T. M. Wanton, L. Specia, MUSST: a multilingual syntactic simplification tool, in: Proceedings of the IJCNLP 2017, System Demonstrations, 2017, pp. 25–28.

[17] Z. Zhu, D. Bernhard, I. Gurevych, A monolingual tree-based translation model for sentence simplification, in: Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), Coling 2010 Organizing Committee, Beijing, China, 2010, pp. 1353–1361. URL: https://www.aclweb.org/anthology/C10-1152.

[18] W. Xu, C. Napoles, E. Pavlick, Q. Chen, C. Callison-Burch, Optimizing statistical machine translation for text simplification, Transactions of the Association for Computational Linguistics 4 (2016) 401–415.

[19] H. Saggion, S. Štajner, S. Bott, S. Mille, L. Rello, B. Drndarevic, Making it simplext: Implementation and evaluation of a text simplification system for spanish, ACM Transactions on Accessible Computing (TACCESS) 6 (2015) 1–36.

[20] G. Barlacchi, S. Tonelli, Ernesta: A sentence simplification tool for children's stories in italian, in: International Conference on Intelligent Text Processing and Computational Linguistics, Springer, 2013, pp. 476–487.

[21] S. Aluísio, C. Gasperin, Fostering digital inclusion and accessibility: the porsimples project for simplification of portuguese texts, in: Proceedings of the NAACL HLT 2010 Young Investigators Workshop on Computational Approaches to Languages of the Americas, 2010, pp. 46–53.

[22] T. Angrosh, Mandaya Nomoto, A. Siddharthan, Lexico-syntactic text simplification and compression with typed dependencies, in: 25th International Conference on Computational Linguistics, 2014.

[23] S. Narayan, C. Gardent, Hybrid simplification using deep semantics and machine translation, in: The 52nd annual meeting of the association for computational linguistics, 2014, pp. 435–445.

[24] X. Zhang, M. Lapata, Sentence simplification with deep reinforcement learning, arXiv preprint arXiv:1703.10931 (2017).

[25] S. Nisioi, S. Štajner, S. P. Ponzetto, L. P. Dinu, Exploring neural text simplification models, in: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Association for Computational Linguistics, Vancouver, Canada, 2017, pp. 85–91. URL: https://www.aclweb.org/anthology/P17-2014. doi:10.18653/v1/P17-2014.

[26] E. Sulem, O. Abend, A. Rappoport, Simple and effective text simplification using semantic and neural methods, in: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Melbourne, Australia, 2018, pp. 162–173. URL: https://www.aclweb.org/anthology/P18-1016. doi:10.18653/v1/P18-1016.

[27] L. Martin, B. Sagot, E. de la Clergerie, A. Bordes, Controllable sentence simplification, arXiv preprint arXiv:1910.02677 (2019).

[28] M. Honnibal, M. Johnson, An improved non-monotonic transition system for dependency parsing, in: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Lisbon, Portugal, 2015, pp. 1373–1378. URL: https://aclanthology.org/D15-1162. doi:10.18653/v1/D15-1162.

[29] E. Sulem, O. Abend, A. Rappoport, BLEU is not suitable for the evaluation of text simplification, in: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Brussels, Belgium, 2018, pp. 738–744. URL: https://www.aclweb.org/anthology/D18-1081. doi:10.18653/v1/D18-1081.

[30] F. Alva-Manchego, L. Martin, A. Bordes, C. Scarton, B. Sagot, L. Specia, ASSET: A dataset for tuning and evaluation of sentence simplification models with multiple rewriting transformations, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Online, 2020, pp. 4668–4679. URL: https://www.aclweb.org/anthology/2020.acl-main.424. doi:10.18653/v1/2020.acl-main.424.

[31] S. Wubben, A. van den Bosch, E. Krahmer, Sentence simplification by monolingual machine translation, in: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Jeju Island, Korea, 2012, pp. 1015–1024. URL: https://www.aclweb.org/anthology/P12-1107.

[32] O. Abend, A. Rappoport, Universal Conceptual Cognitive Annotation (UCCA), in: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics

(Volume 1: Long Papers), Association for Computational Linguistics, Sofia, Bulgaria, 2013, pp. 228–238. URL: https://www.aclweb.org/anthology/P13-1023.

[33] J. P. Kincaid, R. P. Fishburne Jr, R. L. Rogers, B. S. Chissom, Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel, Technical Report, Naval Technical Training Command Millington TN Research Branch, 1975.

[34] K. Papineni, S. Roukos, T. Ward, W.-J. Zhu, Bleu: a method for automatic evaluation of machine translation, in: Proceedings of the 40th annual meeting of the Association for Computational Linguistics, 2002, pp. 311–318.

[35] F. Alva-Manchego, L. Martin, C. Scarton, L. Specia, EASSE: Easier automatic sentence simplification evaluation, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations, Association for Computational Linguistics, Hong Kong, China, 2019, pp. 49–54. URL: https://www.aclweb.org/anthology/D19-3009. doi:10.18653/v1/D19-3009.