I.J. Modern Education and Computer Science, **2018, 1, 57-64**
Published Online January 2018 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2018.01.06

A Novel System for Generating Simple Sentences from Complex and Compound Sentences

Bidyut Das
Department of Information Technology, Haldia Institute of Technology, Haldia-721657, India.
Email: bidyut2002in@gmail.com;

Mukta Majumder
Department of Computer Science and Application, University of North Bengal, Siliguri-734013, India.
Email: mukta_jgec_it_4@yahoo.co.in

Santanu Phadikar
Department of Computer Science and Engineering, Maulana Abul Kalam Azad University of Technology, West Bengal-700064, India.
Email: sphadikar@yahoo.com

Received: 22 October 2017; Accepted: 29 November 2017; Published: 08 January 2018

Abstract—In the field of natural language processing, simple sentence has a great importance; especially for multiple choice question generation, automatic text summarization, opinion mining, machine translation and information retrieval etc. Most of these tasks use simple sentences and include a sentence simplification module as pre-processing or post-processing task. But dedicated tasks for sentence simplification are hardly found. Here we have proposed a novel system for generating simple sentences from complex and compound sentences. Our proposed system is an initiative for simplifying sentence by converting complex and compound sentences into simple ones. Along with this the system classifies the simple sentences of an input corpus from other types of sentences. To generate simple sentences from complex and compound sentences we have proposed a novel algorithm which takes the dependency parsing of the input text and produce simple sentences as output. The experimental result demonstrates that the proposed technique is a promising one.

Index Terms—Sentence simplification, Dependency parsing, Co-reference resolution, Information extraction, Natural language processing.

I. Introduction

Natural language processing (NLP) and machine translation have been studied for decades, but processing human language for extracting meaningful information using computer is still an open area of research; as it was not possible to develop any NLP system or tool which can efficiently mine information from natural language corpora. Though the performance of the available machine translation techniques and NLP tools are quite satisfactory for processing and extracting information from simple sentence but they face major intricacy in the case of complex and compound sentences and as a result degradation in performance [1-3]. This problem occurs due to the embedded dependent clauses, co-references etc. To get better performance from these NLP tools for summarization, frequently asked question answering system, opinion mining and multiple choice question generation, the test corpora are needed to contain simple sentences [4-6]. Hence generating simple sentences from compound and complex sentences is an elementary task for different types of NLP applications and machine translation.

A group of words which make a complete sense or thought is called sentence [7]. A simple sentence is special form with all, the basic property of a sentence. It consists of an independent clause and has a subject and a predicate [8]. It is often short and uncomplicated. A simple sentence is not defined by how short it is [9]. A simple sentence or independent clause is one that has a meaning to a reader or listener [10]. If it does not complete the thought, it may be a dependent clause; by itself, it is a sentence fragment which needs support to stand [11]. Consider the following sentence: 'Anil Dhirubhai Ambani, who married former Bollywood actress, Tina Munim, is the chairman of Reliance Group'.

In the example sentence 'who married former Bollywood actress, Tina Munim' is a sentence fragment or dependent clause. It does not make a complete sense. It needs support from the independent clause 'Anil Dhirubhai Ambani is the chairman of Reliance Group' and the pronoun 'who' is the referent for 'Anil Dhirubhai Ambani'. The above example is a complex sentence. Compound or complex sentence has at least two clauses (one of them is dependent clauses) whereas a simple sentence is made up from one independent clause. Here we have proposed an efficient system for generating simple sentences from complex and compound sentences of an input test corpus. As a pre-processing task we have

Copyright © 2018 MECS

also identified the simple sentences of the input corpus and separated those from rest ones (complex and compound sentences) which are then converted into simple ones by using the proposed technique.

Rest of the article is subdivided as follows. The related previous works are presented in Section 2. Section 3 describes the proposed methodology. Section 4 illustrates the experimental results with proper discussion and the conclusions are included in section 5.

II. Related Works

Sentence simplification is one of the important tasks in natural language processing; having several applications like: frequently asked question generation, text summarization, multiple choice question generation, opinion mining and information retrieval etc [4-6]. But unfortunately we have found in the literature that the sentence simplification task has been unable to achieve amply interest from the researcher. As a consequence the sentence simplification task is limited to a small number of approaches. In this section we have discussed some of the related works which used a module for sentence simplification task.

Chandrasekar et al. observed that the long and complex sentences are barrier for machine translation or automatic parsing. They illustrated that a prior simplification might give better result in automatic analysis of sentences. They considered two alternatives for full parsing which were used for simplification. Their first approach used a Finite State Grammar to generate noun and verb groups whereas the second used a Supper-tagging model to generate dependency linkages. They discussed the impact of these two input representations for the simplification process [12]. Vickrey and Koller used sentence simplification for semantic role labeling. They simplified the sentence, by keeping all arguments of a verb, by removing a little information outside the verb and arguments [2]. Zhu et al. proposed a Tree-based Simplification Model (TSM) to generate the parse tree of a complex sentence by applying splitting, reordering, dropping, and substitution operations. They performed sentence simplification task by tree transformation which was based on the techniques from statistical machine translation (SMT) [13]. Heilman and Smith developed a rule-based algorithm, which was called Simplified Factual Statement Extractor (SFSE) to extract multiple simplified sentences from a source sentence. They showed that adding text simplification improved the result of automatic question generation [14]. Miwa et al. proposed an Entity Focused Sentence Simplification technique for relation extraction. To simplify sentence in relation extraction, the sense of the target sentence is less significant rather than maintaining the truth value of the relation. Hence, they defined two rules, clause selection rules and entity phrase rules. The clause selection rule is used for removing noisy information before and after the relevant clause. Whereas the entity phrase rule is used to simplify an entity holding region without changing the truth value of the relation [15]. Biran et al. presented a method for lexical simplification. Their sentence simplification technique consisted of mainly two stages. The two stages are rule extraction and simplification. In stage one, simplification rules were extracted from the corpora. Each rule consists of an ordered word pair (original and simplified) with a score that indicates the similarity between the words. In stage two, the system decided whether to apply a rule based on the contextual information [16]. Tur et al. presented a sentence simplification method based on dependency parsing. They established its uses to improve intent determination and slot filling tasks in spoken language understanding (SLU) systems. They evaluated their technique using the `Air Travel Information System' (ATIS) corpus with manual and automatic transcriptions. They observed significant error reductions for both in intent determination (30%) and in slot filling (15%) tasks over state-of-the-art techniques [17]. Brouwers et al. presented a method for the syntactic simplification of French texts. The simplification was done in a two-steps process. First, for each sentence, they generated the set of all possible simplifications (over generation step) and then selected the best subset of simplified sentences using several criteria. The sentence over generation module was based on a set of 19 rules which are based on morpho-syntactic features of words and on syntactic relationships within sentences [18]. Tarouti et al. observed that factual question generation from a large complicated sentence was a difficult task in automatic question generation system. They decomposed the source sentence into a set of smaller, simplified sentences and for this task they proposed a method named simplified statement extraction (SSE). They considered a sentence as simplified if it was a declarative sentence (a statement) that could be directly transformed into a question-answer pair without any compression [5]. Nikita et al. proposed a Conditional Random Field (CRF) based approach to identify complex sentences from Punjabi corpus. They provided a framework using CRF to note all accessible features that include the interaction between sentences [19]. Chandni et al. described the different types of sentences present in Punjabi language [20]. Some details of the internal structure of these sentences have also been discussed in this paper.

In this section we have discussed a few tasks which used a sentence simplification module as pre-processing task but we are unable to find any dedicated approach in the literature for generating simple sentences from compound and complex sentences. Our proposed system is an initiative to split complex and compound sentences into simple ones.

III. Proposed Methodology

To generate simple sentences form complex and compound sentences we have taken the help of openly available *'Stanford Parser*[1]*'* and *'Stanford CoreNLP*[2]*'* along with the help of Stanford Typed Dependency

[1] *http://nlp.stanford.edu:8080/parser/index.jsp*
[2] *http://nlp.stanford.edu:8080/corenlp/process*

Manual [21]. The tools are not directly transforming the compound and complex sentences into simple ones. The Stanford Parser provides the dependency parsing of an input sentence and Stanford Deterministic Co-reference Resolution System (module of CoreNLP Suit) helps us to solve co-reference problem. We have proposed an algorithm that works on the dependency parsing [22] of the complex and compound sentences and generate the simple sentences. Before we can apply the proposed algorithm on the dependency parsing of the input corpus, we have to perform a few prepossessing tasks of identifying and separating the simple sentences from others. The proposed technique which contains two distinct phases is presented in Fig 1. The first phase consists of separating simple sentences from the text corpus and the second phase generates the simple sentences from complex and compound sentences.

A. Types of Sentences

Prior to identify simple sentences or generate these from complex or compound sentences; in this regard we need to mention the various types of sentences, that are come across in the input corpora. We have classified them in the following four categories [23].

- **Simple sentence:** A sentence which has only one independent clause and no dependent clauses.
- **Compound sentence:** A sentence which has at least two independent clauses which are combined with coordinating conjunction.
- **Complex sentence:** A complex sentence which has one or more dependent clauses (subordinate clauses). A dependent clause cannot stand alone. Therefore, a complex sentence must also have at least one independent clause. These clauses are combined by using subordinate conjunction.
- **Compound-Complex sentence:** A sentence which has two or more independent clauses and one or more dependent clauses.

B. Identifying simple sentences

To separate the simple sentences from other type of sentences we have analyzed the dependency structures of the input sentences that are generated by Stanford Parser. First of all, we have counted the number of 'nsubj' or 'nsubjpass' from the dependencies of an input sentence. The 'nsubj' and 'nsubjpass' are categorized as subject according to 'Stanford Typed Dependency Manual' [24]. In a simple sentence only one 'nsubj' or 'nsubjpass' is there. Hence a sentence contains more than one 'nsubj' or 'nsubjpass' is considered as other than simple sentence.

C. Generating simple sentences from complex and compound sentences

To generate the simple sentences from compound and complex sentences we have used Modified Stanford Dependency (MSD) structure which is derived from Basic Stanford Dependency (BSD) and Collapsed Stanford Dependency (CSD) structures. The MSD is created by taking the 'nsubj' or 'nsubjpass' from CSD and the remaining parts from BSD. The ten dependencies 'acl', 'appos', 'advcl', 'cc', 'ccomp', 'conj', 'dep', 'mark', 'parataxis' and 'ref' are omitted while creating the MSD as shown in Table 1 and Table 2.

D. Simple Sentence Generation (SSG) Algorithm

Input: Text 'T'.
Output: Set of Simple Sentences.
Begin:
T = { S_1, S_2 , S_3, S_4, ... S_z } where S_1, S_2, S_3, S_4,…. S_z are the input sentences of Text 'T'.

For each sentence S_i (i = 1; i < z; i++) {

Step 1: Create Basic and Collapsed SD of S_i using Stanford Parser.

Step 2: Generate Modified SD (MSD) of S_i by taking subject clauses, 'nsubj' or 'nsubjpass' from Collapsed SD and other linked parts from Basic SD.

Step 3: Remove dependencies, marked as 'acl', 'appos', 'advcl', 'cc', 'ccomp', 'conj', 'dep', 'mark', 'parataxis' and 'ref'.

Step 4: Identify subject clause 'nsubj' or 'nsubjpass'.

Step 5: Traverse all the parts that have links to each subject clause ('nsubj' or 'nsubjpass') and find the linked words.

Step 6: Rearrange the words found in Step 5 to generate the complete simple sentences.

} **End for**
End

For an explanation of the SSG algorithm, we consider the following two example sentences.

'Sachin Tendulkar, who is a MP of Indian Parliament, played cricket'. (Complex sentence)

'MS Dhoni is the captain of Indian Cricket Team and he is the Vice-President of India Cement' (Compound sentence).

A simple sentence is build up of one independent clause; on the other hand compound or complex sentence is build from at least two clauses [8]. Therefore, the complex and compound sentences are split into clauses that can form simple sentences. We have found the dependency parsing of the first example sentence (complex sentence) from Stanford CoreNLP suit (which includes Stanford Parser) to split it into simple sentences. From the sample sentence, we get the parse result in Stanford Dependency (SD) notations as shown in Table1.

A clause can be identified from MSD which category is subject, e.g. 'nsubj', or 'nsubjpass'. Now the basic clause can be extracted from head as subject part. From the aforementioned dependencies, there are two basic clauses: 'Tendulkar MP' and 'Tendulkar played'. As soon as we get basic clause, we can add other parts to make our clause a complete and meaningful sentence. Once we get the pair, i.e. nsubj (MP-7, Tendulkar-2); we get all the dependencies that have links to it, except any dependency which category is subject, then extract unique word from these dependencies. Based on example, nsubj (MP-7, Tendulkar-2); if we start traversing from Tendulkar-2, we get the following dependencies: compound (Tendulkar-2, Sachin-1).

Copyright © 2018 MECS

Next we traversed from MP-7 and we get the followings dependencies:

cop(MP-7, is-5)

det(MP-7, a-6)

nmod(MP-7, Parliament-10)

From the result above, we get new dependencies to traverse (i.e. find another dependencies that have ‘is-5’, ‘a-6’, ‘Parliament-10’ in either head or dependent).

Now these dependencies are

case (Parliament-10, of-8)

compound (Parliament-10, Indian-9)

In this step, we have finished traversing all dependencies linked to nsubj (MP-7, Tendulkar-2).

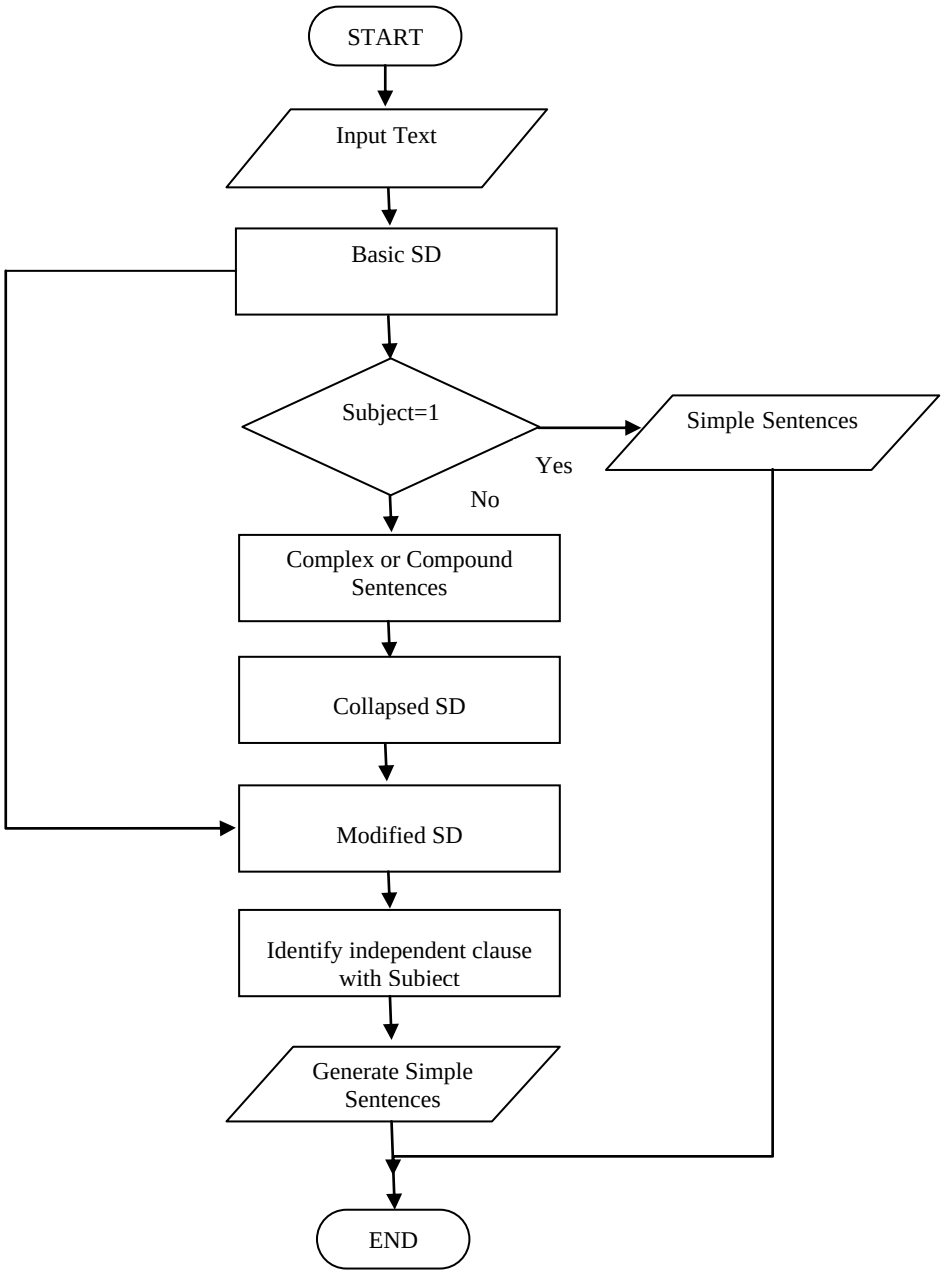

Fig.1. A Graphical representation of the proposed technique

Table 1. Dependencies of the First sentence

Basic Stanford Dependencies (BSD)	Collapsed Stanford Dependencies (CSD)	Modified Stanford Dependencies (MSD)
compound(Tendulkar-2, Sachin-1)	*compound(Tendulkar-2, Sachin-1)*	*nsubj(MP-7, Tendulkar-2)*
nsubj(played-12, Tendulkar-2)	*nsubj(MP-7, Tendulkar-2)*	*nsubj(played-12, Tendulkar-2)*
nsubj(MP-7, who-4)	*nsubj(played-12, Tendulkar-2)*	*compound(Tendulkar-2, Sachin-1)*
cop(MP-7, is-5)	*ref(Tendulkar-2, who-4)*	*cop(MP-7, is-5)*
det(MP-7, a-6)	*cop(MP-7, is-5)*	*det(MP-7, a-6)*
acl:relcl(Tendulkar-2, MP-7)	*det(MP-7, a-6)*	*case(Parliament-10, of-8)*
case(Parliament-10, of-8)	*acl:relcl(Tendulkar-2, MP-7)*	*compound(Parliament-10, Indian-9)*
compound(Parliament-10, Indian-9)	*case(Parliament-10, of-8)*	*nmod(MP-7, Parliament-10)*
nmod(MP-7, Parliament-10)	*compound(Parliament-10, Indian-9)*	*root(ROOT-0, played-12)*
root(ROOT-0, played-12)	*nmod:of(MP-7, Parliament-10)*	*dobj(played-12, cricket-13)*
dobj(played-12, cricket-13)	*root(ROOT-0, played-12)*	
	dobj(played-12, cricket-13)	

Table 2. Dependencies of the second sentence

Basic Stanford Dependencies (BSD)	Collapsed Stanford Dependencies (CSD)	Modified Stanford Dependencies (MSD)
compound(Dhoni-2, MS-1)	*compound(Dhoni-2, MS-1)*	*nsubj(captain-5, Dhoni-2)*
nsubj(captain-5, Dhoni-2)	*nsubj(captain-5, Dhoni-2)*	*nsubj(Vice-President-14, he-11)*
cop(captain-5, is-3)	*cop(captain-5, is-3)*	*compound(Dhoni-2, MS-1)*
det(captain-5, the-4)	*det(captain-5, the-4)*	*cop(captain-5, is-3)*
root(ROOT-0, captain-5)	*root(ROOT-0, captain-5)*	*det(captain-5, the-4)*
case(Team-9, of-6)	*case(Team-9, of-6)*	*root(ROOT-0, captain-5)*
amod(Team-9, Indian-7)	*amod(Team-9, Indian-7)*	*case(Team-9, of-6)*
compound(Team-9, Cricket-8)	*compound(Team-9, Cricket-8)*	*amod(Team-9, Indian-7)*
nmod(captain-5, Team-9)	*nmod:of(captain-5, Team-9)*	*compound(Team-9, Cricket-8)*
cc(captain-5, and-10)	*cc(captain-5, and-10)*	*nmod(captain-5, Team-9)*
nsubj(Vice-President-14, he-11)	*nsubj(Vice-President-14, he-11)*	*cop(Vice-President-14, is-12)*
cop(Vice-President-14, is-12)	*cop(Vice-President-14, is-12)*	*det(Vice-President-14, the-13)*
det(Vice-President-14, the-13)	*det(Vice-President-14, the-13)*	*case(Cement-17, of-15)*
conj(captain-5, Vice-President-14)	*conj:and(captain-5, Vice-President-14)*	*compound(Cement-17, India-16)*
case(Cement-17, of-15)	*case(Cement-17, of-15)*	*nmod(Vice-President-14, Cement-17)*
compound(Cement-17, India-16)	*compound(Cement-17, India-16)*	
nmod(Vice-President-14, Cement-17)	*nmod:of(Vice-President-14, Cement-17)*	

Next, we have extracted the words from all head and dependent, and then arranged the words in ascending order based on number appended to these words. This numbers are indicating the word orders in original sentence. In this way we get the following sentence:

'Sachin Tendulkar is a MP of Indian Parliament'.

Similarly, we get the second sentence 'Sachin Tendulkar played cricket'.

We have applied the same approach for the second sentence (compound sentence). The dependency structure of the second sentence is shown in Table 2. From the MSD, we search for 'nsubj' or 'nsubjpass' which is categorized as subject and get the two basic clauses:

'Dhoni captain' and 'He Vice-President'. Once we get the pair, i.e. nsubj (captain-5, Dhoni-2); we get all dependencies that have links to it. Based on the head dependency nsubj (captain-5, Dhoni-2); we start traversing from captain-5 and we get the following dependencies:

cop(captain-5, is-3)

det(captain-5, the-4)

nmod(captain-5, Team-9)

Next, we traversed from Dhoni-2 and we get the dependency:

compound (Dhoni-2, MS-1)

From the result above, we get new dependencies to traverse (i.e. find another dependencies that have 'is-3', 'the-4', 'Team-9' in either head or dependent). Now the linked dependencies are:

case(Team-9, of-6)

amod(Team-9, Indian-7)

compound(Team-9, Cricket-8)

In this step, we have finished traversing all dependencies linked to nsubj (captain-5, Dhoni-2). Next, we extract words from all head and dependent parts, and arrange them in ascending order based on the number appended to these words. In this way we get the following sentence:

'MS Dhoni is the captain of Indian Cricket Team'. Similarly we get the other sentence: 'He is the Vice-President of India Cement'.

Here 'He' refers to 'MS Dhoni'. Therefore, we get the two simple sentences from the above compound sentence.

IV. Results and Discussion

To test the accuracy of the proposed system we have collected the data from different openly available online sources. As there is no standard for computing the performance of such kind of system, we have taken the judgments of five human linguistic experts on the correctness of simplification of the sentences and consider the accuracy as the average of their judgments.

To calculate the accuracy of the system we have extracted the text from two Wikipedia pages namely, '2011 Cricket World Cup' and '2015 Cricket World Cup'. These two pages contain a total of about 165 sentences (Table 3); out of them 73 sentences are identified as simple. Hence 92 sentences are given to the system which

Copyright © 2018 MECS

generates 236 simple sentences. These sentences are checked by 5 human evaluators. According to them 208, 219, 211, 215 and 222 sentences respectively are accurate simple sentences. Therefore the system's accuracy is 91.102%. Table 4 summarizes the results of our proposed system which generates simple sentences from compound and complex sentences. From the evaluation score given in Table 4, it is understood that the proposed system is able to generate the quality simple sentences from compound and complex sentences.

The only drawback of our system is that, it sometime generates incomplete sentences. An incomplete sentence, or sentence fragment, is a group of words that does not form a complete sense or meaning. A complex sentence must have one or more dependent clauses with at least one independent clause. A sentence having one dependent clause without any independent clause is one example of such incomplete sentence which is generated quite a few times while the system splits the complex sentences. To overcome this we have removed those sentences that end with 'verb' using POS tagging [24-25] as a post-processing task. For example, the sentence:

'New Zealand government had also assured that the Zimbabwean team would be allowed to take part in the tournament' generates two simple sentences.

'New Zealand government had also assured' and 'the Zimbabwean team would be allowed to take part in the tournament'. The first sentence is incomplete and ends with verb 'assured'. In the post-processing phase we have filtered such type of sentences.

V. Conclusion

Simple sentences are used in different Natural Language Processing applications for extracting meaningful information. A novel and efficient system is presented in this paper for not only categorizing simple sentences from others but also generating simple sentences from complex and compound sentences. For classifying simple sentence the proposed technique analyzes the parse structures of the input corpus and consider the ones as simple sentences which are having a single 'nsubj' or 'nsubjpass' (subject). The dependency structures of rest of the sentences (other than simple sentences) are fed into the proposed Simple Sentence Generating (SSG) Algorithm to generate simple sentences from complex or compound sentences. The experimental results indicate that the proposed technique is efficient and effective. Now, this system can be used in different NLP applications like: multiple choice question generation, summarization, opinion mining and information retrieval etc. We have considered automatic Multiple Choice Question preparation by generating simple sentences from complex and compound sentences as the future direction of our research.

Table 3. Used Data Set

	Wikipedia page name	
	2015_Cricket_World_Cup	2011_Cricket_World_Cup
Web Source	https://en.wikipedia.org/wiki/2015_Cricket_World_Cup	https://en.wikipedia.org/wiki/2011_Cricket_World_Cup
Sentences	78	87
Simple Sentences	30	43
Complex and Compound Sentences	48	44

Table 4. Performance of simple sentence generation

Complex and Compound Sentences	Simple Sentences	Correct Simple Sentences (Evaluators Judgment)	Accuracy (%)
92	236	Evaluator 1: 208 Evaluator 2: 219 Evaluator 3: 211 Evaluator 4: 215 Evaluator 5: 222	91.102

References

[1] S. Wubben, A.V.D. Bosch, E. Krahmer, "Sentence simplification by monolingual machine translation," *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, Long Papers-Volume 1, pp. 1015-1024, 2012.

[2] D. Vickrey, D. Koller, "Sentence simplification for semantic role labeling," *Proceedings of the Association for Computational Linguistics*, pp. 344-352, 2008.

[3] C. Poornima, V. Dhanalakshmi, K.M. Anand, K.P. Soman, "Rule based sentence simplification for english to tamil machine translation system", *International Journal of Computer Applications* vol. 25(8), pp. 38-42, 2011.

[4] A. Bawakid, M. Oussalah, "Sentences simplification for automatic summarization," *Proceedings of IEEE 10th International Conference of Cybernetic Intelligent Systems (CIS), IEEE*, pp. 59-64, 2011.

[5] F.A. Tarouti, J, Kalita, C. McGrory, "Sentence simplification for question generation," *Proceedings of the International Conference on Computing and Communication Systems*, 2015, http://www.cs.uccs.edu/ jkalita/papers/2015/AltaroutiFerasI3CS2015.pdf

[6] M. Majumder, S.K. Saha, "A system for generating multiple choice questions: With a novel approach for sentence selection," *ACL (Association for Computational Linguistics), IJCNLP*, p. 64, 2015.

[7] T.B. McArthur, F. McArthur, "The Oxford Companion to the English Language, Oxford Companions Series,"

Oxford University Press, 1992, https://books.google.co.in/books?id=yIoYAAAAIAAJ
[8] B. Backman, "Building Sentence Skills: Tools for Writing the Amazing English Sentence," *Teacher Created Resources, Incorporated: Middle School Series*, 2003, https://books.google.co.in/books?id=n-0wXZf4In4C
[9] G. Lutz, D. Stevenson, "The Writer's Digest Grammar Desk Reference," *F+W Media*, 2005, https://books.google.co.in/books?id=SsQ9ugnMcpUC
[10] F. Obrecht, "Minimum Essentials of English. Barron's Educational Series, 1999, https://books.google.co.in/books?id= j4yE7y5c NoC
[11] T.P. Klammer, R.M. Shultz, A.D. Volpe, "Analyzing English Grammar," *Pearson Education*, India, 2007
[12] R. Chandrasekar, C. Doran, B. Srinivas, "Motivations and methods for text simplification," *Proceedings of the 16th Conference on Computational linguistics (Association for Computational Linguistics*), vol. 2, pp. 1041-1044, 1996
[13] Z. Zhu, D. Bernhard, I. Gurevych, "A monolingual tree-based translation model for sentence simplification," *Proceedings of the 23rd International Conference on Computational Linguistics (Association for Computational Linguistics*), pp. 1353-1361, 2010.
[14] M. Heilman, N.A. Smith, "Extracting simplified statements for factual question generation," *Proceedings of QG2010: the Third Workshop on Question Generation*, pp. 11-20, 2010.
[15] M. Miwa, R. Satre, Y. Miyao, J.I. Tsujii, "Entity-focused sentence simplification for relation extraction," *Proceedings of the 23rd International Conference on Computational Linguistics (Association for Computational Linguistics)*, pp. 788-796, 2010.
[16] O. Biran, S. Brody, N. Elhadad, "Putting it simply: a context-aware approach to lexical simplification," *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, Short papers-vol. 2, pp. 496-501, 2011.
[17] G. Tur, D. Hakkani-Tur, L. Heck, S. Parthasarathy, "Sentence simplification for spoken language understanding," *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE*, pp. 5628-5631, 2011.
[18] L. Brouwers, D. Bernhard, A.L. Ligozat, T. Francois, "Syntactic sentence simplification for French," *Proceedings of the 3rd Workshop on Predicting and Improving Text Readability for Target Reader Populations (PITR), EACL*, pp. 47-56, 2014.
[19] S.K.D. Nikita, S.K. Sharma, "Detection of complex sentences in Punjabi language using CRF," *Journal of Innovation in Electronics and Communication Engineering* vol. 5(2), pp. 42-46, 2015.
[20] Chandni, R. Narula, S.K. Sharma, "Identification and separation of simple, compound and complex sentences in Punjabi language," *International Journal of Computer Applications & Information Technology*, vol. 6, pp.123-128, 2014.
[21] M.C. De Marneffe, C.D. Manning, "Stanford typed dependencies manual," Technical report, *Stanford University*, 2008, https://nlp.stanford.edu/software/dependencies manual.pdf
[22] R. Kokare, K. Wanjale, "A Natural Language Query Builder Interface for Structured Databases Using Dependency Parsing," *International Journal of Mathematical Sciences and Computing*, vol. 1(4), pp. 11-20, 2015.
[23] P. Atteberry, "Sentence Types," http://www.pitt.edu/atteberr/comp/0150/grammar/sentencetypes.html
[24] B. Santorini, "Part-of-speech tagging guidelines for the penn treebank project (3rd revision)", 1990.
[25] R. Khoury, "Sentence Clustering Using Parts-of-Speech," *International Journal of Information Engineering and Electronic Business*, vol. 4(1), pp.1-9, 2012.

Authors' Profiles

Bidyut Das is an Asstistant Professor of the Haldia Institute of Technology, Haldia, India. He was born on 8th February, 1982 in India. He received his B.Sc. and M.Sc. degrees in Computer Science from Vidyasagar University, West Bengal, India, in 2002 and 2004 respectively. He did his M.Tech in Information Technology from School of Information Technology, West Bengal University of Technology, Kolkata, India in 2008. He received Gold Medal in M.Sc and Silver Medal in M.Tech. His research interests include Natural Language Processing, Text Mining, Machine Learning, Pattern Recognition, Information Retrieval and Human Computer Interaction.

Mukta Majumder is an Assistant Professor in the department of Computer Science and Application, University of North Bengal, Siliguri, West Bengal, India. He has submitted his Ph.D thesis in the Department of Computer Science and Engineering, Birla Institute of Technology Mesra, Ranchi, India. He has completed his post graduation from National Institute of Technical Teachers Training and Research's, Kolkata, India and graduation from Jalpaiguri Government Engineering College, Jalpaiguri, India. He has served Vidyasagar University as an Assistant Professor of Computer Centre from 22nd December 2014 to 17th October 2017. After that he is serving this University. His research interests include Natural Language Processing, Machine Learning, ICT Based Teaching-Learning, Microfluidic System, and Biochip.

Dr. Santanu Phadikar is currently working as an Associate Professor in the department of Computer Science and Engineering, Maulana Abul Kalam Azad University of Technology, West Bengal, India. He has passed his graduation from Vidyasagar University and completed his master degrees from University of Calcutta. He has received his Ph.D degree from Bengal Engineering and Science University, Shibpur, West Bengal, India. He has received University Gold Medal for securing highest marks in gradation in his subject. He has also received National Scholarship for higher study. He has served Vidyasagar University as a Lecture of Computer Science from 2002 to 2006. After that he is serving this University. He is working in the field of Smart Farming, Natural Language Processing, Machine Learning, Voice Processing and Image Processing etc. He is currently running the project ISEA-II as Chief Investigator.

Copyright © 2018 MECS

How to cite this paper: Bidyut Das, Mukta Majumder, Santanu Phadikar, "A Novel System for Generating Simple Sentences from Complex and Compound Sentences", International Journal of Modern Education and Computer Science(IJMECS), Vol.10, No.1, pp. 57-64, 2018.DOI: 10.5815/ijmecs.2018.01.06

Copyright © 2018 MECS