2019 4th International Conference on Information Systems and Computer Networks (ISCON)
GLA University, Mathura, UP, India. Nov 21-22, 2019

Sentence Simplification using Syntactic Parse trees

Avishek Garain
Computer Science and Engineering
Jadavpur University
avishekgarain@gmail.com

Arpan Basu
Computer Science and Engineering
Jadavpur University
arpan0123@gmail.com

Rudrajit Dawn
Computer Science and Engineering
Jadavpur University
rudrajitdawn@gmail.com

Sudip Kumar Naskar
Computer Science and Engineering
Jadavpur University
sudip.naskar@gmail.com

***Abstract*—Text simplification is one of the domains in Natural Language Processing which offers great promise for exploration. Simplifying sentences offer better results, as compared dealing with complex/compound sentences, in many language processing applications as well. Recently, Neural Networks have been used in simplifying texts, be it by state of the art LSTM's and GRU cells or by Reinforcement learning models. In contrast, in this work, we present a classical approach consisting of two separate algorithms, for simplification of complex and compound sentences to their corresponding simple forms.**

Keywords- Syntactic Parse tree, SBAR, BLEU, Anytree

I. Introduction

On a trivial basis, a complex or a compound sentence can be differentiated from simple sentences through the presence of conjunctions (coordinating or subordinating).

A compound sentence can be defined as an amalgamation of two or more different independent clauses based on a similar idea. The independent clauses can be joined by a coordinating conjunction (for, and, nor, but, or, yet, so) or by a semicolon, as we can see in the compound sentence examples below.

- *She did not cheat on the test, for it was the wrong thing to do.*
- *I really need to go to work, but I am too sick to drive.*
- *The sky is clear; the stars are twinkling.*

On the other hand, a complex sentence is similar to a compound sentence, but varying in the fact that it consists of at least one dependent clause, related to a independent clause. An independent clause has the ability to stand alone as a sentence. It always makes a complete thought. A dependent clause cannot stand alone, even though it has a subject and a verb. Lets take a look at some examples of complex sentences.

- *I was snippy with him because I was running late for work.*
- *Because I was running late for work, I was snippy with him.*

Sentence simplification is necessary for dealing with problems related with long sentences. These include embedded clauses such as relative clause. To overcome this problem, the long sentence is simplified into smaller sentences. Sentence simplification can be used as a pre-processing tool in several applications such as Natural Language Processing, Query Processing and Speech Processing. In Text Summarization, sentence simplification is used to shorten the original Text without losing the meaning of the content. Also, it has been proved that training Machine Translation systems, with simple sentences only, lead to better translation outputs [7].

Recently, many splitting technique has been developed for English language, mainly employing the usage of Neural Networks, be it by state of the art LSTM's and GRU cells or by Reinforcement learning models. In this work, we have developed a classical, rule based approach, that can simplify English sentences, from complex and compound sentences. Since, the work is based on simple syntactic rules, the system is fast ans modular as well.

The rest of the paper has been organized as follows. Section II will specify the state-of-the-art that has been developed so far in this domain. Section III will define the data, on which the work has been done. This will be followed by the methodology of developing our prototype in Section IV. The paper will end with the results and concluding remarks of the work in Section V and VI, respectively.

II. Related Work

Zhang and Lapata [14], proposed a model called Deep Reinforcement Sentence Simplification, where they use an encoder-decoder model along with Deep Reinforcement framework to carry on the task of simplification and reported good accuracy scores.

Vu et al. [13], in their work used an architecture with augmented memory capabilities named Neural Semantic Encoders and have reported Bleu scores as high as 92.02. The model gives unrestricted access to the entire source sequence (complex sentence) stored in the memory and thus, through attention model, gives importance to each and every word of the sentence keeping priorities for arrangement too.

Guo et al. [3] introduced a sequence to sequence based learning model with improvement using paraphrasing capabilities via multi-task learning. They proposed a multi-level layered soft sharing approach where each auxiliary task shares different level layers of the sentence simplification model, depending on the tasks semantic versus lexico-syntactic nature.

Poornima et al. [10] used rule based Sentence Simplification on their English to Tamil Machine Translation System.

They made use of dependency parse tree for the method of simplification. It is interesting to note that they have reported significance increase in accuracy due to simplification of Complex and Compound sentences to simple sentences and also decrease in error rates. This report is promising for the immediate usage of our algorithm in several other fields in Natural Language Processing and validating variation in performance.

Jonnalagadda et al. [4] presented an effective summarization system on basis of text simplification and results were quite promising.

V et al. [11] used the strategies like a extractive summarization system which is focussed on tasks, simplification of sentences, and lexical expansion of topic words for increasing content responsiveness.

III. Data

As such no such state of the art corpus consisting of aligned data on Complex and Compound sentences to corresponding Simple sentences is available to test on for our algorithm. So we developed our own dataset by collecting Compound and Complex sentences from various digital resources. Hence, 10,000 sentences were collected using this method, where 5200 were complex sentences and 4800 were compound sentences.

IV. Methodology

Our sentence simplification algorithm consists of two parts; first part pertaining to simplification of compound sentences which where clauses are connected using coordinating conjunctions, and the second part pertains to simplification of compound sentences which where clauses are connected using subordinating conjunctions. The collected sentences after POS tagging by NLTK[6] POS tagger, were shallow parsed using Stanford Parser[8], before hand, to obtain the parse tree structure. For both the simplification module, the algorithms works on the same parse tree structure, obtained earlier. Structure of a parse tree for sample complex and compound sentences are shown in Figure 1 and 2, respectively.

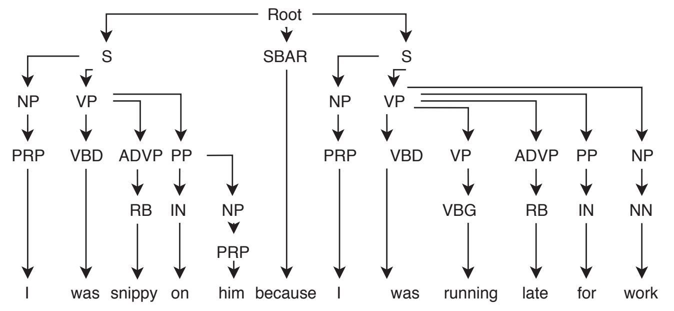


Fig. 1: Complex sentence parse tree sample.

A. Simplification of Compound Sentences

1) Outline: The first part of the algorithm is mainly for splitting a sentence based on the coordinating conjunctions present in the parse tree. This type of sentences generally contain a `CC` tag present in them. The algorithm takes advantage of this tag and the corresponding syntactic parse-tree to

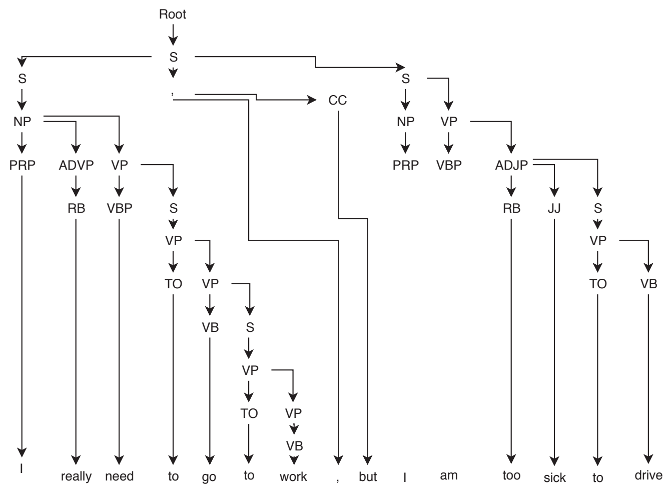


Fig. 2: Compound sentence parse tree sample.

split the constructions joined together by the conjunction. The algorithm is recursively applied to split sentences into simple sentences, till no `CC` tag remains.

2) Algorithm: For the input sentence, initially, a syntactic parse-tree is generated. Thereafter, the node corresponding to the `CC` tag is found. Let it be called **X**. Then we find the parent of the parent of the node with the tag. Let it be called **Y**. For all nodes which are siblings of **X**, we attach each such node to node **Y** to get distinct parse-trees. The tag and the related punctuation (mainly commas) are removed.

Each generated parse-tree is a simplified sentence devoid of the conjunction which was being considered. We note that each parse-tree is not necessarily a correct parsing; we only use them to construct the simplified sentences. The above approach is then applied recursively on each simple sentence thus produced. The algorithm stops when there are no remaining conjunctions.

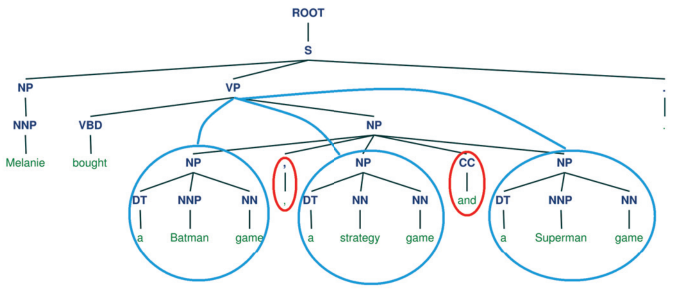


Fig. 3: Figure representing the conjunction removal algorithm

Figure 3 pictorially represents the conjunction removal algorithm. The blue links are the newly formed links, each blue circle forming an independent sentence. The red circles denote the parts being discarded.

TABLE I: Table showing the simple sentences generated on an example input

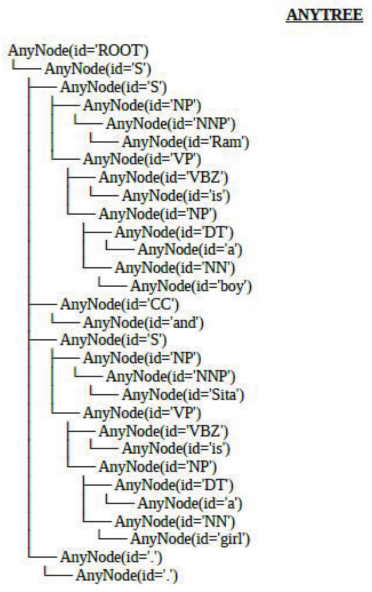
Input sentence
Melanie bought a Batman game, a strategy game and a Superman game.

Output simple sentences
Melanie bought a Batman game .
Melanie bought a strategy game .
Melanie bought a Superman game .

B. *Simplification of Complex Sentences*

1) Outline: We feed a complex sentence to our algorithm as input. Then it will return a list of sentences without `SBAR` tag in any of the sentences . All of the sentences may not be simple . Only `SBAR` part will be removed and equivalent simpler sentences will be returned .

2) Algorithm: First the shallow parsed tree is converted to ANYTREE[1]. This is done because of easy handability of ANYTREE inside code. Example of ANYTREE parse tree is shown in Figure 4.



```
AnyNode(id='ROOT')
└── AnyNode(id='S')
    ├── AnyNode(id='S')
    │   ├── AnyNode(id='NP')
    │   │   └── AnyNode(id='NNP')
    │   │       └── AnyNode(id='Ram')
    │   └── AnyNode(id='VP')
    │       ├── AnyNode(id='VBZ')
    │       │   └── AnyNode(id='is')
    │       └── AnyNode(id='NP')
    │           ├── AnyNode(id='DT')
    │           │   └── AnyNode(id='a')
    │           └── AnyNode(id='NN')
    │               └── AnyNode(id='boy')
    ├── AnyNode(id='CC')
    │   └── AnyNode(id='and')
    ├── AnyNode(id='S')
    │   ├── AnyNode(id='NP')
    │   │   └── AnyNode(id='NNP')
    │   │       └── AnyNode(id='Sita')
    │   └── AnyNode(id='VP')
    │       ├── AnyNode(id='VBZ')
    │       │   └── AnyNode(id='is')
    │       └── AnyNode(id='NP')
    │           ├── AnyNode(id='DT')
    │           │   └── AnyNode(id='a')
    │           └── AnyNode(id='NN')
    │               └── AnyNode(id='girl')
    └── AnyNode(id='.')
        └── AnyNode(id='.')
```

Fig. 4: Example of Anytree parse tree

Initially, the subtree rooted at `SBAR` node is detected. If the `SBAR` part contains `NP`, then sentences can be made inside `SBAR` part. Some examples of the same is shown below.

Example:
Because I was late, I became angry.

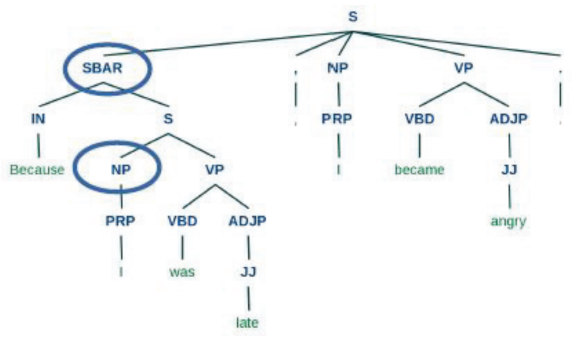


Fig. 5: Syntactic parse tree

If the `SBAR` part does not contain `NP` then `NP` and `VBZ` from non-`SBAR` part is copied to `SBAR` part to make sentences inside `SBAR` part. This is shown in Figure 5.

Example:
While eating food Ram is singing a song.

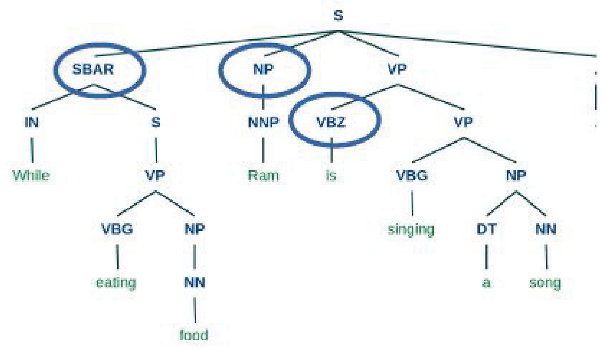


Fig. 6: Syntactic parse tree

If `CC` is inside `SBAR` then different sub-sentences **(S)** inside `SBAR` are considered as different `SBAR`'s and previous points are applied to each of the `SBAR`'s. This is shown in Figure 6.

Example:
While eating food and drinking water Ram is singing a song .

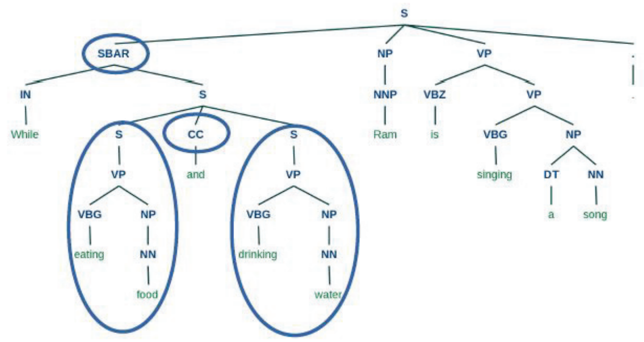


Fig. 7: Syntactic parse tree

[1]https://anytree.readthedocs.io/en/latest/index.html

One sentence from non-SBAR part can be made. This is shown in Figure 7.

Example:
If he comes I will go. Parsing of this is shown in Figure 8.

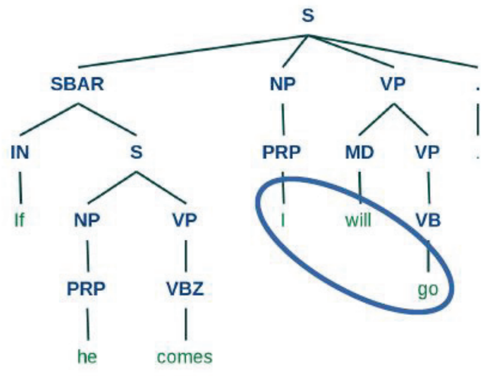


Fig. 8: Syntactic parse tree

Some more parse trees fitting perfectly in our algorithm are as follows:

Complex sentence: After Shyam came , Ram went .
Simple sentences: ['Ram went .', 'Shyam came .']

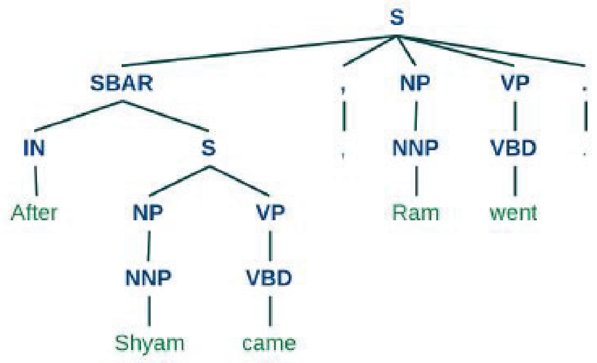


Fig. 9: Syntactic parse tree

Complex sentence: After she ate the cake , Emma visited Tony in his room .
Simple Sentences: ['Emma visited Tony in his room .', 'she ate the cake .']

V. Results

Our system was tested using manual and automatic evaluation metrics. For the manual evaluation part, two human annotators, A and B, fluent in English, were asked to classify the results of our algorithm into two classes; correct and incorrect. The inter annotator agreement for both complex and compound sentences are shown in Table II and III. The **S**tandard **E**rror (SE) for the agreement ranges between 0.009-0.010, which is very promising.

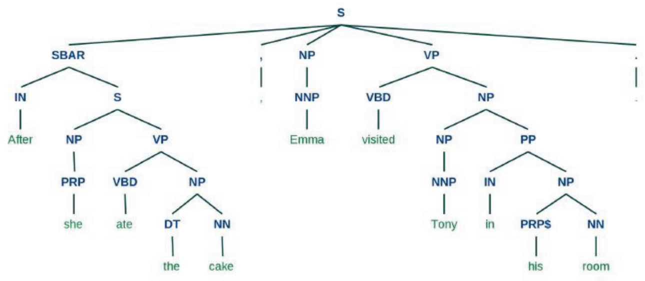


Fig. 10: Syntactic parse tree

TABLE II: Inter Annotator agreement for complex sentences,

		Annotator B	
		Correct	**Incorrect**
Annotator A	**Correct**	4798	20
	Incorrect	30	352
Kappa	0.929		

For the automated evaluation part, the annotators, manually converted 100 complex sentences and 100 compound sentences to simple sentences. The same set of complex and compound sentences were converted by our algorithm, as well. The simple sentences produced by our algorithm, were then compared with the manually converted ones. The automatic evaluation metric used here was BLEU [9]. BLEU, or the Bilingual Evaluation Understudy, is a score for comparing a candidate text to one or more reference texts. BLEU score came as 0.89 for complex sentences and 0.91 for compund sentences. This, again, is highly promising.

Also, statistics of complex and compound sentences when converted to simple sentences, is given in Table IV.

VI. Conclusion

Our approach is a different approach as compared to the current practice of using deep learning, and hence has large scope of improvement and modifications as a part of future work. Since the algorithm is mainly dependent on the syntactic parse tree generated, so wrong parse trees may hinder the efficiency of our algorithm. Correct parse trees as seen above gives perfect results. In the meantime, we came to identify simple sentences to consist of rule: NP$\rightarrow$VP. These simple sentences are easier for other Language Processing systems to handle, and may help in increasing their efficiency in feature extraction and accuracy. This work is left to be discovered as part of future work.

Acknowledgment

The authors would like to thank Mr. Sainik Kumar Mahata for his help and suggestions in organizing the paper. We would also like to thank Mr. Sourav Kumar Mandal, who inspired us with his mathematical prowess, to carry on this work and give it a proper direction.

References

[1] C P, Dhanalakshmi V, Kumar M, Kp S (2011) Rule based sentence simplification for english to tamil machine

TABLE III: Inter Annotator agreement for compound sentences,

		Annotator B	
		Correct	Incorrect
Annotator A	Correct	4296	26
	Incorrect	32	446
Kappa	0.932		

TABLE IV: Statistics of converted sentences.

	Complex	Compound
	5,200	4,800
Simple	9,800	10,200

translation system. International Journal of Computer Applications 25, DOI 10.5120/3050-4147

[2] Coster W, Kauchak D (2011) Simple english wikipedia: A new text simplification task. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2, Association for Computational Linguistics, Stroudsburg, PA, USA, HLT ’11, pp 665–669, URL http://dl.acm.org/citation.cfm?id=2002736.2002865

[3] Guo H, Pasunuru R, Bansal M (2018) Dynamic multi-level multi-task learning for sentence simplification. CoRR abs/1806.07304, URL http://arxiv.org/abs/1806.07304, 1806.07304

[4] Jonnalagadda S, Tari L, Hakenberg J, Baral C, Gonzalez G (2009) Towards effective sentence simplification for automatic processing of biomedical text. In: Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Companion Volume: Short Papers, Association for Computational Linguistics, Stroudsburg, PA, USA, NAACL-Short ’09, pp 177–180, URL http://dl.acm.org/citation.cfm?id=1620853.1620902

[5] Li T, Li Y, Qiang J, Yuan YH (2018) Text Simplification with Self-Attention-Based Pointer-Generator Networks: 25th International Conference, ICONIP 2018, Siem Reap, Cambodia, December 1316, 2018, Proceedings, Part V, pp 537–545. DOI 10.1007/978-3-030-04221-9_48

[6] Loper E, Bird S (2002) Nltk: The natural language toolkit. In: Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1, Association for Computational Linguistics, Stroudsburg, PA, USA, ETMTNLP ’02, pp 63–70, DOI 10.3115/1118108.1118117, URL https://doi.org/10.3115/1118108.1118117

[7] Mahata SK, Mandal S, Das D, Bandyopadhyay S (2018) Smt vs nmt: A comparison over hindi & bengali simple sentences. arXiv preprint arXiv:181204898

[8] Manning CD, Surdeanu M, Bauer J, Finkel J, Bethard SJ, McClosky D (2014) The Stanford CoreNLP natural language processing toolkit. In: Association for Computational Linguistics (ACL) System Demonstrations, pp 55–60, URL http://www.aclweb.org/anthology/P/P14/P14-5010

[9] Papineni K, Roukos S, Ward T, Zhu WJ (2002) Bleu: a method for automatic evaluation of machine translation. In: Proceedings of the 40th annual meeting on association for computational linguistics, Association for Computational Linguistics, pp 311–318

[10] Poornima C, Dhanalakshmi V, Anand K, Soman K (2011) Rule based sentence simplification for english to tamil machine translation system. International Journal of Computer Applications 25(8):38–42

[11] V L, Suzuki H, Brockett C (2006) Microsoft research at duc2006: Task-focused summarization with sentence simplification and lexical expansion

[12] Vickrey D, Koller D (2008) Sentence simplification for semantic role labeling. In: Proceedings of ACL-08: HLT, Association for Computational Linguistics, Columbus, Ohio, pp 344–352, URL https://www.aclweb.org/anthology/P08-1040

[13] Vu T, Hu B, Munkhdalai T, Yu H (2018) Sentence simplification with memory-augmented neural networks. CoRR abs/1804.07445, URL http://arxiv.org/abs/1804.07445, 1804.07445

[14] Zhang X, Lapata M (2017) Sentence simplification with deep reinforcement learning. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Copenhagen, Denmark, pp 584–594, DOI 10.18653/v1/D17-1062, URL https://www.aclweb.org/anthology/D17-1062