Exploring the effects of Sentence Simplification on Hindi to English Machine Translation System

Kshitij Mishra **Ankush Soni** **Rahul Sharma** **Dipti Misra Sharma**
Language Technologies Research Centre
IIIT Hyderabad
{kshitij.mishra,ankush.soni,rahul.sharma}@research.iiit.ac.in,
dipti@iiit.ac.in

Abstract

Even though, a lot of research has already been done on Machine Translation, translating complex sentences has been a stumbling block in the process. To improve the performance of machine translation on complex sentences, simplifying the sentences becomes imperative. In this paper, we present a rule based approach to address this problem by simplifying complex sentences in Hindi into multiple simple sentences. The sentence is split using clause boundaries and dependency parsing which identifies different arguments of verbs, thus changing the grammatical structure in a way that the semantic information of the original sentence stay preserved.

1 Introduction

Cognitive and psychological studies on 'human reading' state that the effort in reading and understanding a text increases with the sentence complexity. Sentence complexity can be primarily classified into 'lexical complexity' and 'syntactic complexity'. Lexical complexity deals with the vocabulary practiced in the sentence while syntactic complexity is governed by the linguistic competence of native speakers of a particular language. In this respect, the modern machine translation systems are similar to humans. Processing complex sentences with high accuracy has always been a challenge in machine translation. This calls for automatic techniques aiming at simplification of complex sentences both lexically and syntactically. In context of natural language applications, lexical complexity can be handled significantly by utilizing various resources like lexicons, dictionary, thesaurus etc. and substituting infrequent words with their frequent counterparts. However, syntactic complexity requires mature endeavors and techniques.

Machine Translation systems when dealing with highly diverges language pairs face difficulty in translation. It seems intuitive to break down the sentence into simplified sentences and use them for the task. Phrase based translation systems exercise a similar approach where system divides the sentences into phrases and translates each phrase independently, later reordering and concatenating them into a single sentence. However, the focus of translation is not on producing a single sentence but to preserve the semantics of the source sentence, with a decent readability at the target side.

We present a rule based approach which is basically an improvement on the work done by (Soni et al., 2013) for sentence simplification in Hindi. The approach adapted by them has some limitations since it uses verb frames to extract the core arguments of verb; there is no way to identify information like time, place, manner etc. of the event expressed by the verb which could be crucial for sentence simplification. A parse tree of a sentence could potentially address this problem. We use a dependency parser of Hindi for this purpose. (Soni et al., 2013) didn't consider breaking the sentences at finite verbs while we split the sentences on finite verbs also.

This paper is structured as follows: In Section 2, we discuss the related work that has been done earlier on sentence simplification. Section 3 addresses criteria for classification of complex sentences. In section 4, we discuss the algorithm used for splitting the sentences. Section 5 outlines evaluation of the systems

Proceedings of the Workshop on Automatic Text Simplification: Methods and Applications in the Multilingual Society, pages 21–29,
Dublin, Ireland, August 24th 2014.

using both BLEU scores and human readability . In Section 6, we conclude and talk about future work in this area.

2 Related Work

Siddharthan (2002) presents a three stage pipelined approach for text simplification. He has also looked into the discourse level problems arising from syntactic text simplification and proposed solutions to overcome them. In his later works (Siddharthan, 2006), he discussed syntactic simplification of sentences. He has formulated the interactions between discourse and syntax during the process of sentence simplification. Chandrasekar et al. (1996) proposed Finite state grammar and Dependency based approach for sentence simplification. They first build a stuctural representation of the sentence and then apply a sequence of rules for extracting the elements that could be simplified. Chandrasekar and Srinivas (1997) have put forward an approach to automatically induce rules for sentence simplification. In their approach all the dependency information of a words is localized to a single structure which provides a local domain of influence to the induced rules.

Sudoh et al. (2010) proposed divide and translate technique to address the issue of long distance reordering for machine translation. They have used clauses as segments for splitting. In their approach, clauses are translated separately with non-terminals using SMT method and then sentences are reconstructed based on the non-terminals. Doi and Sumita (2003) used splitting techniques for simplifying sentences and then utilizing the output for machine translation. Leffa (1998) has shown that simplifying a sentence into clauses can help machine translation. They have built a rule based clause identifier to enhance the performance of MT system.

Though the field of sentence simplification has been explored for enhancing machine translation for English as source language, we don't find significant work for Hindi. Poornima et al. (2011) has reported a rule based technique to simplify complex sentences based on connectives like subordinating conjunction, relative pronouns etc. The MT system used by them performs better for simplified sentences as compared to original complex sentences.

3 Complex Sentence

In this section we try to identify the definition of sentence complexity in the context of machine translation. In general, complex sentences have more than one clause (Kachru, 2006) and these clauses are combined using connectives. In the context of machine translation, the performance of system generally decreases with increase in the length of the sentence (Chandrasekar et al., 1996). Soni et al. (2013) has also mentioned that the number of verb chunks increases with the length of sentence. They have also mentioned the criteria for defining complexity of a sentence and the same criteria is apt for our purpose also. We consider a sentence to be complex based on the following criteria:

- Criterion1 : Length of the sentence is greater than 5.
- Criterion2 : Number of verb chunks in the sentence is more than 1.
- Criterion3 : Number of conjuncts in the sentence is greater than 0.

Table 1 shows classification of a sentence based on the possible combinations of 3 criteria mentioned above.

4 Sentence Simplification Algorithm

We propose a rule based system for sentence simplification, which first identifies the clause boundaries in the input sentence, and then splits the sentence using those clause boundaries. Once different clauses are identified, they are further processed to find shared argument for non-finite verbs. Then, the Tense-Aspect-Modality(TAM) information of the non-finite verbs is changed. Below example (12) illustrates the same,

Table 1: Classification of a sentence as simple or complex

Criterion1	Criterion2	Criterion3	Category
No	No	No	Simple
No	No	Yes	Simple
No	Yes	No	Simple
No	Yes	Yes	Simple
Yes	No	No	Simple
Yes	No	Yes	Complex
Yes	Yes	No	Complex
Yes	Yes	Yes	Complex

(1) raam ne khaanaa khaakara pani piya
Ram food after+eating water drink+past
'Ram drank water after eating.'

We first mark the boundaries of clauses for example (12). 'raam' and 'khaanaa' are starts, and 'khaakara' and 'piya' are ends of two different clauses respectively. Once the start and end of clauses are identified we break the sentence into those clauses. So for above example, the two clauses are:

1. 'raam ne pani piya'

2. 'khaanaa khaakara'

Once we have the clauses, we post process those clauses which contain non-finite verbs, and add the shared argument and TAM information for these non-finite clauses. After post-processing, the two simplified clauses are:

1. 'raam ne pani piya.'

2. 'raam ne khaanaa khaayaa.'

4.1 Algorithm

Our system comprises of a pipeline incorporating various modules. The first module determines the boundaries of clauses (clause identification) and splits the sentence on the basis of those boundaries. Then, the clauses are processed by a gerund handler - which finds the arguments of gerunds, shared argument adder which fetches the shared arguments between verbs, TAM(Tense Aspect Modality) generator which changes the TAM of other verbs on the basis of main verb. The figure 4.1 shows the data flow of our system, components of which have been discussed in further detail in this section.

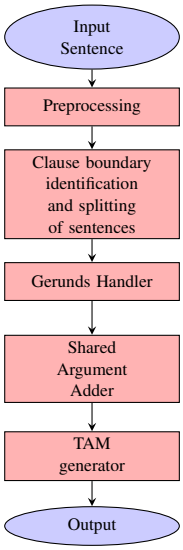

Figure 1: Data Flow

4.1.1 Preprocessing

In this module, raw input sentences are processed and each lexical item is assigned a POS tag, chunk and dependency relations information in SSF format(Bharati et al., 2007; Bharati et al., 2009). We have used (Jain et al., 2012) dependency parser for preprocessing. Example (2) shows the output of this step.
Input sentence:

(2) raam ne khaanaa khaayaa aur paani piyaa.
Ram+erg food eat+past and water drink+past
'Raam ate food and drank water'

Output: Figure (1) shows the different linguistic information in SSF format. Tag contains the Chunk and POS information of the sentence, and drel in feature structure stores different dependency relations in a sentence.

Offset	Token	Tag	Feature structure
1	((	NP	<fs name='NP' drel='k1:VGF'>
1.1	raama	NNP	<fs af='raama,n,m,sg,3,d,0,0'>
1.2	ne	PSP	<fs af='ne,psp,,,,,,'>
	))		
1 2	((	NP	<fs name='NP2' drel='k2:VGF'>
2.1	khaanaa	NN	<fs af='khaanaa,n,m,sg,3,d,0,0' name='khaanaa'>
	))		
3	((	VGF	<fs name='VGF' drel='ccof:CCP'>
3.1	khaayaa	VM	<fs af='KA,v,m,sg,any,,yA,yA' name='khaayaa'>
	))		
4	((	CCP	<fs name='CCP'>
4.1	aur	CC	<fs af='Ora,avy,,,,,,' name='aur'>
	))		
5	((	NP	<fs name='NP3' drel='k2:VGF2'>
5.1	paani	NN	<fs af='pAnI,n,m,sg,3,d,0,0' name='paani'>
	))		
6	((	VGF	<fs name='VGF2' drel='ccof:CCP'>
6.1	piyaa	VM	<fs af='pIyA,unk,,,,,,' name='piyaa'>
	))		

Figure 1: SSF representation for example 2

4.1.2 Clause boundary Identification and splitting of sentences

This module takes the input from preprocessing module and identifies the clause boundaries in the sentence. Once clause boundaries are identified, the sentence is divided into different clauses. We have used the technique mentioned in Sharma et al. (2013) which has shown how implicit clause information present in dependency trees/relations can be used to extract clauses from a sentence. Once we mark the clause boundaries using this approach, we break the sentence into different simple clauses along those clause boundaries. The example(3) given below illustrates the same.

(3) raam jisne khaanaa khaayaa ghar gayaa
Ram who+rel. food eat+past home go+past
'Ram who ate food, went home'

Example(3) with clause boundaries marked is, (raam (jisne khaanaa khaayaa) ghar gayaa). Once the clause boundaries are marked, we break the sentence using those boundaries. So for Example(3), split clauses are,

1. raam ghar gayaa.
2. jisne khaanaa khaayaa.

4.1.3 Gerunds Handler

Since, Sharma et al. (2013) identifies clause boundary for non-finite and finite verb only, gerunds are not handled in the previous module. This module is used to handle gerunds in the given sentence. In this module, the gerund chunks are first indentified and then further processed after getting the arguments. Consider an example:

(4) logon ko sambodhit karne ke baad dono netaon ne pradhanmantri ko istifa saunpa
people address doing after both leaders Prime minister to resignation gave
'After addressing people, both leaders gave resignation to the prime minister'

In the above example, the clause boundary identifier module marks the entire sentence as a clause but *karne ke baad* is a gerund chunk (verb chunk) here, which is marked as VGNN according to the tagset of the POS tagger used. According to definition of complex sentence given in section 3 gerunds also introduce complexity in a sentence. Therefore, in order to simplify such sentences, we use dependency parsing information for extracting the arguments of gerund and splitting the sentence.

Here *logon ko* and *sambodhit* are the arguments of verb chunk *karne ke baad*. Here *ke baad* is postposition of verb *karne* so, *ke baad* is splitted from *karne* and it has been used with the pronoun *is* to make the sentence more readable.

1. *logon ko sambodhit karne*
2. *iske baad dono netaon ne pradhanmantri ko istifa saunpa*

4.1.4 Shared Argument Adder

After identifying clauses and handling gerunds, the shared arguments are identified between the verbs and sentences are formed accordingly. For example:

(5) (ram (chai aur paani peekar) soyaa)
(ram (tea and water after drinking) slept)
'ram slept after drinking tea and water'

Here *ram* is the shared argument(k1-karta) of both the verbs *peekar* and *soyaa* . The dependency parser used, marks the inverse dependencies for shared arguments which helps in . So the output of this module is:

1. *ram chai aur paani peekar.*
2. *ram soyaa.*

4.1.5 TAM generator

The split sentences given by the above module are converted into more readable sentences using this module. The form of other verbs is changed using TAM information of the main verb provided by the morph, as shown in Figure 1. For example:
INPUT:

1. *ram chai aur paani peekar.*
2. *ram soyaa.*

OUTPUT:

1. *ram chai aur paani peeyaa.*
2. *ram soyaa.*

Here *soyaa* is the main verb having *yaa* as TAM. Word generator[1] has been used to generate the final verb given root form of the verb and TAM of the verb. Here *pee* is the root form of *peekar* and *yaa* is given as the TAM. Word generator generates *peeyaa* as the final word which is used in the sentence.

5 Evaluation

We have taken a corpus of 100 complex sentences for the evaluation of our tool. These sentences were taken from the Hindi treebank (Bhatt et al., 2009; Palmer et al., 2009). Evaluation of both sentence simplification and its effects on google MT system for Hindi to English(google translate) was performed. The evaluation of sentence simplification is a subjective task which considers both readability and preservation of semantic information. Hence both manual as well as automatic evaluations have been performed.

5.1 Automatic Evaluation

We have used BLEU score (Papineni et al., 2002) for automatic evaluation of both tasks; sentence simplification and enhancing MT system. Higher the BLEU score, closer the target set is to the reference set. The maximum attainable value is 1 while minimum possible value is 0. For our Automatic evaluation we adopted the same technique as Specia (2010) using BLEU metric. We have achieved 0.6949 BLEU score for sentence simplification task. For MT system, we have evaluated the system with and without sentence simplification tool. It was observed that the system with sentence simplification tool achieved 0.4986 BLEU score whereas the system without sentence simplification gave BLEU score of 0.4541.

5.2 Human Evaluation

To ensure the simplification quality, manual evaluation was also done. 20 sentences were randomly selected from the testing data-set of 100 sentences. Output of these 20 sentences, from the target set were manually evaluated by 2 subjects, who have done basic courses in linguistics, for judging 'Readability' and 'Simplification' quality on the scale of $0-3$, 0 being the worst and 3 being the best.
For Simplification performance, scores were given according to following criteria :

[1]Taken from the ILMT pipeline.

- 0 = None of the expected simplifications performed.
- 1 = Some of the expected simplifications performed.
- 2 = Most of the expected simplifications performed.
- 3 = Complete Simplification.

After taking input from all the participants the results averaged out to be 2.5.
For Human evaluation of MT system, the subjects had to select the better translation between system with sentence simplification tool and system without it. The subjects reportedly observed a better translation of the system with sentence simplification tool. It was reported that 12 out of 20 sentences were translated better after being simplified, and quality of 3 remained unchanged.

Translation quality of 5 was reported to be better before simplification. This happened because the system breaks the sentences at every verb chunk it encounters, which in some cases makes the sentence lose its semantic information.

For example the sentence below contains five verb chunks. The system breaks the sentence into five sentences.:

(6) *yah poochne par ki kya we dobaara congress mein lautenge sangama ne kaha ki na*
this ask on that what he again Congress in return Sangama told that neither
to iski zarurat hai aur na hi peeche lautane ka sawal hi uthta hai
then its requirement is and nor back return question raises is
'On asking whether he would return again in Congress, Sangma replied that neither there is need of this nor there is the question of reverting back.'

System's Output

1. (7) kya we dobaara congress mein lautenge
what he again Congress in return
'Would they return again in Congress ?'

2. (8) yah poochane par sangama ne kaha
this ask on Sangama told
'On asking this, Sangama said.'

3. (9) na to iski zarurat hai
neither its requirement is told
'Neither it is needed.'

4. (10) na hee peeche lautana hai
neither back return is
'Neither he will return.'

5. (11) iska sawal uthta hai
its question raises is
'The question arises.'

It is clearly observable that the simplified sentences failed to preserve the meaning of the original sentence. Further, the system does not change the *vibhakti* (Bharati et al., 1995) of the simplified sentences which, in some cases makes the sentence lose its meaning. For example

(12) machharon ke katne ke baad wo beemar hue
Mosquitoes of bite of after they sick became
'They became sick after being bitten by the mosquitoes.'

System's Output:

1. (13) machharon ke kata
   Mosquitoes of bite
   '*Not a valid sentence*'

2. (14) is ke baad wo beemar hue
   this of after they sick became
   'After this they became sick.'

In the first simplified sentence the *vibhakti "ke"* should have been changed to *"ne"* for the formation of a valid sentence.

6 Conclusion and Future Work

As shown in the results, after simplifying the sentences, BLEU score of the translation increases by 4.45. The manual evaluation also got encouraging results in simplification and readability with a score of **2.5** on a scale of $0-3$. There is a clear indication that our tool can enhance the performance of MT for complex sentences by simplifying them. Future work will include minimizing the lose of semantic information while splitting the sentences and making simplified sentences more readable and grammatically correct. In addition to extending the system, evaluating the impact of our tool on other NLP tasks like parsing, dialog systems, summarisation, question-answering systems etc. is also a future goal.

Acknowledgements

We would like to thank Riyaz Ahmad Bhat, Rishabh Shrivastava and Prateek Saxena for their useful comments and feedback which helped us to improve this paper and Anshul Bhargava, Arpita Batra, Abhijat Sharma, Gaurav Kakkar, Jyoti Jha and Urmi Ghosh for helping us in annotation.

References

Akshar Bharati, Vineet Chaitanya, Rajeev Sangal, and KV Ramakrishnamacharyulu. 1995. *Natural language processing: a Paninian perspective*. Prentice-Hall of India New Delhi.

Akshar Bharati, Rajeev Sangal, and Dipti M Sharma. 2007. Ssf: Shakti standard format guide. pages 1–25.

Akshara Bharati, Dipti Misra Sharma, Samar Husain, Lakshmi Bai, Rafiya Begam, and Rajeev Sangal. 2009. Anncorra: Treebanks for indian languages, guidelines for annotating hindi treebank.

R. Bhatt, B. Narasimhan, M. Palmer, O. Rambow, D.M. Sharma, and F. Xia. 2009. A multi-representational and multi-layered treebank for hindi/urdu. In *Proceedings of the Third Linguistic Annotation Workshop*, pages 186–189. Association for Computational Linguistics.

Raman Chandrasekar and Bangalore Srinivas. 1997. Automatic induction of rules for text simplification. *Knowledge-Based Systems*, 10(3):183–190.

Raman Chandrasekar, Christine Doran, and Bangalore Srinivas. 1996. Motivations and methods for text simplification. In *Proceedings of the 16th conference on Computational linguistics-Volume 2*, pages 1041–1044. Association for Computational Linguistics.

Takao Doi and Eiichiro Sumita. 2003. Input sentence splitting and translating. In *Proc. of Workshop on Building and Using Parallel Texts, HLT-NAACL 2003*, pages 104–110.

Naman Jain, Karan Singla, Aniruddha Tammewar, and Sambhav Jain, 2012. *Proceedings of the Workshop on Machine Translation and Parsing in Indian Languages*, chapter Two-stage Approach for Hindi Dependency Parsing Using MaltParser, pages 163–170. The COLING 2012 Organizing Committee.

Yamuna Kachru. 2006. *Hindi*, volume 12. John Benjamins Publishing.

Vilson J Leffa. 1998. Clause processing in complex sentences. In *Proceedings of the First International Conference on Language Resources and Evaluation*, volume 1, pages 937–943.

M. Palmer, R. Bhatt, B. Narasimhan, O. Rambow, D.M. Sharma, and F. Xia. 2009. Hindi Syntax: Annotating Dependency, Lexical Predicate-Argument Structure, and Phrase Structure. In *The 7th International Conference on Natural Language Processing*, pages 14–17.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

C Poornima, V Dhanalakshmi, Anand M Kumar, and KP Soman. 2011. Rule based sentence simplification for english to tamil machine translation system. *International Journal of Computer Applications*, 25(8):38–42.

Rahul Sharma, Soma Paul, Riyaz Ahmad Bhat, and Sambhav Jain. 2013. Automatic clause boundary annotation in the hindi treebank.

Advaith Siddharthan. 2002. An architecture for a text simplification system. In *Language Engineering Conference, 2002. Proceedings*, pages 64–71. IEEE.

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

Ankush Soni, Sambhav Jain, and Dipti Misra Sharma. 2013. Exploring verb frames for sentence simplification in hindi. Proceedings of the Sixth International Joint Conference on Natural Language Processing, pages 1082–1086. Asian Federation of Natural Language Processing.

Lucia Specia. 2010. Translating from complex to simplified sentences. In *Computational Processing of the Portuguese Language*, pages 30–39. Springer.

Katsuhito Sudoh, Kevin Duh, Hajime Tsukada, Tsutomu Hirao, and Masaaki Nagata. 2010. Divide and translate: improving long distance reordering in statistical machine translation. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 418–427. Association for Computational Linguistics.