Grammar Checker for Hindi and Other Indian Languages

Anjani Kumar Ray, Vijay Kumar Kaul
Center for Information and Language Engineering
Mahatma Gandhi Antarrashtriya Hindi Vishwavidyalaya, Wardha (India)

Abstract: Grammar checking is one of the most widely used techniques within natural language processing (NLP) applications. Grammar checkers check the grammatical structure of sentences based on morphological and syntactic processing. These two steps are important parts of any natural language processing systems. Morphological processing is the step where both lexical words (parts-of-speech) and non-word tokens (punctuation marks, made-up words, acronyms, etc.) are analyzed into their components. In syntactic processing, linear sequences of words are transformed into structures that show grammatical relationships among the words in the sentence (Rich and Knight 1991) and between two or more sentences joined together to make a compound or complex sentence. There are three main approaches/models which are widely used for grammar checking in a language; probability-based models, rule-based models and hybrid models. In rule-based grammar checking, each sentence is completely parsed to check whether the sentence is grammatically well-formed. In absence of the potential syntactic parsing approach, incorrect or not-so-well grammatically formed sentences are analyzed or produced. The preset paper is an exploratory attempt to devise the hybrid models to identify the grammatical relations and connections of the words to phrases to sentences to the extent of discourse. Language Industry demands such economic programmes doing justice and meeting the expectations of language engineering.

Keywords: Grammar Checking, Language Engineering, Syntax Processing, POS Tagging, Chunking, morphological Analysis

Introduction: Grammar Checker is an NLP application that helps the user to write correct sentence in the concerned language. It compiles the give text in such a manner that reports the error found after analyzing the text at the grammatical parameters of the language. Grammar checker enables you to correct the most unnoticeable mistakes/errors. Grammar

Checker helps the user to write better Hindi with efficiently equipped features that corrects the texts. Grammar Checker corrects grammatical mistake based on the context of complete sentences with unmatched accuracy. For the utilization of Grammar checking software the user has to enter the text that is intended to be corrected for grammar check and punctuation mistakes the user has to just click to check error related to Grammar and punctuation and it will report the error found in the word or phrase with a wavy underline. To get the possible solution to error then you have to click right hand on it will show the possible solution for particular error. Most of mistake which are committed by the user be it related to Gender, number and/or related to agreement between phrases of sentence. Most of Grammar checkers have the spell checker embedded in it.

State of Art: Grammar checking is one of the most widely used techniques within natural language processing (NLP) applications. Grammar checker checkes the grammatical structure of sentence based on morphological and syntactic processing. These two steps are important parts of any natural language processing systems. Morphological processing is the step where both lexical words (parts-of-speech) and non-word tokens (punctuation marks, made-up words, acronyms, etc.) are analyzed into their components. In syntactic processing, linear sequences of words are transformed into structures that show grammatical relationships among the words of a given sentence.

Approaches/models and other frameworks, widely used for grammar checking in a language are:

- Probability Based Models,
- Rule Based Models
- Hybrid Models.

Methodology: The methodology is used in designing the grammar checker is deductive. System can extract the information related to grammar rules: morphological, syntactic and semantic evaluation of each word in a given sentence. The information cannot get in one go but the information is extracted in each phase of the system and the information is accumulated and stored with each individual word. The grammar checker has been built on the basis of syntactic rules the Hindi language observes. System converts the document into token of paragraph. The paragraph is further divided into sentences and each sentence is divided into token of word and non-word sequences. Using these techniques we are in a position to know about each token which has its original belonging to any given paragraph and sentence.

Tokenization Phase: In the process of tokenization the text input divided in long string of characters, into subunits, called tokens[18] . Here the token means an individual appearance of a word in certain position in a given text. For instance one can consider (लड़कों ləɽəkoň ladakon) as an instance of word (लड़का ləɽəka ladaka).

In a running text the token are generally separated by white space but token words may contain following characters as punctuation markers and other as part of the word for e.g.

लड़का लड़का' लड़का,'
लड़का" लड़का. लड़का).

ləɽəka ləɽəka' ləɽəka,' ləɽəka"
ləɽəka. ləɽəka).

ladakaa ladakaa' ladakaa,' ladakaa"
ladakaa. ladakaa).

Abbreviation Identification: In Tokenization process, we have to be very carefull about using of dot (.) in defining sentence boundaries and it is also used in writting abbreviations. There is a need for writing the rules to identify abbreviations.

We can predict some abbreviations such as the abbreviations are written with dot (.) and space such as अ. ग. प. is used as abbreviations for, असम गण परिषद (əsəm gəɳ pərɪʂəd̪ asama gan parishad). In this, a sequence of letter-dot-letter-dot pattern is appearing. Abbreviations are written with dot (.) and without space between alphabets such as, अ.ग.प. is used as abbreviations for असम गण परिषद (əsəm gəɳ pərɪʂəd̪ asama gan parishad)

abbreviations are written without space and without dot (.) such as अगप Is used as abbreviations for असम गण परिषद (əsəm gəɳ pərɪʂəd̪ asama gan parishad).

Some abbreviation are written as a group of characters without space and without dot (.) and such word also belongs to lexicon of Hindi language such as आप i.e. used as abbreviations for आम आदमी पार्टी (aama aadamee paartee) but as we know that आप is a very prompt Hindi pronoun as well. This type of ambiguity cannot be resolved by one sentence.

Name Entity Recognition (NER): NER plays an important role in identifying the token word which represents one identity i.e. NER and it behaves like a Proper Noun. Thus NER identification process will affect overall performance of the system. For NER identification we discuss some rules which helps in identifying the NER and combine group of token which is part of any NER then system can be treat as a one single unit or single token.

Some Rules are as follows …

Identification of Name of Person

In Hindi language, common title that is used in writing the names of people such as

श्री श्रीमान श्रीमति,डॉ. इंजि. मोहतरमा, मैडम, मैम, etc.

ʃri ʃriman ʃrimət̪ɪ,ɖɒ. ɪnʤɪ. moɦət̪ərəma, mɛɖəm, mɛm

When the system encounters such title word it checks the next word which may be the part of name of a Person written in fashion

[Title] [First Name] [Middle Name] [Sur Name] [Honorific Marker word]

Identification of Date String

Date string is being identified by using regular expression and is consider the best way to do it. There are several patterns for writing the date string such as

01/02/2017, 1/2/2017, 01/02/17, 1//2/17, 01-Feb-2017, Feb 01, 2017 etc.

During the identification process of data and time string, we have to write several regular expressions to identify all types of date and time string.

1. Identification of Numeric Value / Currency

There are several words in a document like 200 रु, रु 200, रु 200/- these word will represent for some currency value. Its POS category will be "NCD" i.e. numerical cardinal.

On other hand, some words like 9 वा, 9 वे, 9 वी. These words contain two tokens. The first token is a numerical value and the second token which is attached to previous token gives some additional information related to Gender, Number etc. such as when token "वी" is added to a numerical value the word become a feminine gender and likewise when token "वे" is added to any numerical token then word formed by these tokens having depict the plural Number.

Therefore when we analyze the tokens we have to extract some information about gender, Number and sometimes about the Person. This information is store with word. This information will help in deducting the problem or error in the Phrase that contains these types of tokens.

Morphological Analysis : A Morphological Analyzer is a tool which takes a word as input and produces linguistic information (lexical form) of the word, such as its class, tense, etc. which is required by all Natural Language Processing Systems. In the proposed system Morphological Analysis is being used for analyzing the word and its lexical form. Morphological Analysis of word will give some information about its Gender and Number and in case the word is a verb it will give information about Tense, Aspect and Modality (TAM).

There are various approaches used in designing the Morphological Analysis i.e. Corpus Based Approach, Stemming Approach, Paradigm Approach and Finite State Transducer (FST) Based approaches. In the proposed System Paradigm Approach for Morphological Analysis is used, as Hindi is a highly inflectional language. Hence, the Paradigm based approach is most efficient approach for morph analyzer for Hindi and probably for other Indian languages since a root word can generate various words by adding the suffix, prefix, or circumfix.

This type of Morphological Analysis is done in two Phases

Nominal Word Morphology

Verb Identification using Morphological Analysis

POS Tagging : Part-of-speech tagging is a process of assigning the part-of-speech or other lexical class marker to each word in the corpus. Tags for natural language are much more ambiguous. Part of speech tagger plays an important role in the speech recognition, natural language parsing and information retrieval[13]. In tagging algorithm, string words are taken as input and algorithm select appropriate tag from tagset for the word and assign it to the word. The Part of Speech can broadly be divided in two super categories i.e. closed class and open class. Members

of Closed class type of any language may be fixed. Close classes of language may differ from language to language. Some of close classes in Hindi are as follows:

Postpositions	को,ने,के,का,की,द्वारा,से,में,पर,अनुरूप,अनुसार, खिलाफ,जरिए,तरफ,तहत,तौर,दौरान,दौलत,बनिस्बत,बाद ko,ne,ke,ka,ki,d̪wara,se,men,pər,ənʊrup,ənʊsar,kʰɪlapʰ,ʤərɪe,t̪ərəpʰ,t̪əɦət̪,t̪ər,d̪ɔran, bəd̪ɔlət̪,bənɪsbət̪,bad̪ ko,ne,ke,kaa,kee,dvaaraa,se,men,para,anuroopa,anusaara,khilaapha,jarie,tarapha,tahata,taura,dauraan,badaulata, banisbata,baad
Determiners/ Quantifiers	ज्यादा,तनिक,तमाम,थोड़ा, अति,अथाह,अधूरी,अधिक,अधिकाधिक अधिकांश,अपर्याप्त,अपार,अरसे,अरसा, अव्वल ʤjad̪a,t̪ənɪk,t̪əmam,ʈʰoɽa, əʈ̪ɪ,əʈ̪ʰaɦ,əd̪ʱuri,əd̪ʱɪk,əd̪ʱɪkad̪ʱɪk, əd̪ʱɪkanʃ, əpərjapt̪, əpar, ərəse,ərəsa, əwwəl jyaadaa,tanika,tamaama,thodaa, ati,athaaha,adhree,adhika,adhikaadhika adhikaansha,aparyaapta,apaara,arase, arasaa, avvala
Conjunctions	अथवा,एवं,और,तथा,या,व əʈ̪ʰəwa,enw,ər,t̪əʈ̪ʰa,ja,w athava, evm,aur,tatha,yaa,vaa
Auxiliary Verbs	रहा/रहे है/हैं रही है/हैं रहा था रही थी/थीं रहे थे/थें rəɦa/rəɦe ɦɛ/ɦɛn rəɦi ɦɛ/ɦɛn rəɦa ʈ̪ʰa rəɦi ʈ̪ʰi/ʈ̪ʰin rəɦe ʈ̪ʰe raha/rahe hei/hein rahi hai/hein raha tha rahi thi/thi rahe the/then
Particles	अलावा,केवल,तक,बजाय,भर,भी,ही ələwa,kewəl,t̪ək,bəʤaj,bʱər,bʱi,ɦi alava, kewal, tak, bajaay, bhar, bhi,hi
Numerals	एक, दो, हजार, लाख, करोड़ ek, d̪o, ɦəʤar, lakʰ, kəroɖ ek, do, hjaar, laakh, karor

Automatically assigning tag to each word is not trivial.

Verb Identification: To identify whether the given word is verb or not is according to following suffix table. Word is broken into root word and suffix. The system will check whether the root word belongs to diction of root and suffix. If it matches to the suffix table then we can say that the given word is a verb and system will accumulate its function along with the word. These information will help to check grammar associated with noun phrase which behaves like a subject in a sentence.

Table Verb Suffix

Suffix	Function
ंुगी/ंूगी/ुंगी/ूंगी/ेंगी/ंॅगी/ुँगी/यीं/एँगी/गी/एगी/गीं/ईं/एगीं/येंगी/ंगीं/ँगीं/ऊँगी jin/ẽgi/gi/egi/gin/in/egin/jengi/-----/----/ũgi	Tense – Future Number – Plural Gender – Feminine
तीं/ीं t̪iŏ/in	Tense – Past Number – Plural Gender – Feminine
वातीं/वाईं/वायीं/वानी/ातीं/ाइ/ाईं wat̪in/wain/wajin/wani/aatin/aai/aaie	Double Causative Verb, Number – Plural Gender – Feminine
इए/इएगा/िएगा/ं/एं/यें/ेंं/एँ/जिए ɪe/ɪega/---/---/ en/jeŏ/--/ ẽ/ʤɪe	Number – Plural Honorific Marker : Yes

Table Verb Suffix

Suffix	Function
ंुगा/ंूगा/ुंगा /ूंगा/ेंगा/ूँ गा/ुँगा/ऊँगा /ोगे/एँगे/एगा/ेंगे/ँगे/एंगें/एंगे/ऊँ/यें गा/उंगा/ ूँ/ंगे/गे ũga/engen/enge/ ũ/jenga/ʊnga/ge	Tense – Future Number – Plural Gender – Mescaline
ते/ा/या/े/या/ए/ ना/ला/ो/रना/यो/ य//ंू/ूं	Number – Singular
ेगी/ओगी/एगी//ये गी/गा/ये/िए/ओगी /एंगी/एँगी ogi/engi/ẽgi	Tense – Future Number – Singular Gender – Feminine
ती /ाती/यी//ली/ी/ इ/ई/यी ṯi/aati/yi/li/i/i/ee/ yi	Tense – Present Number – Singular Gender – Feminine
नी/ानी/वानी /ायी वायी/वाई/वाती/यी /लायी/लाई ni/ani/wani/ayi/w ayi/waii/wati/yii/ laayi/laae	Tense – VCT Number – Singular Gender – Feminine
लवाई/लवायी ləwai/ləwaji lavaai/lavaayi	Tense – VDC Number – Singular

Table Verb Suffix

Suffix	Function
	Gender – Feminine
ेगा/एगा/येगा/गा ega/jaga/ga	Tense – Future Number – Singular Gender – Mescaline
ता ṯa	Tense – Present Number – Singular Gender – Mescaline
ओ/ए /ाए//ाये/आ o/e/aye/aye/aa	Tense – Past Number – Singular Gender – Mescaline
वाना/वाता/वाया/ वाते/ाया/ाते/ये/ या/ाना/वाना/वा या/वाने/वा/लवाया/ वाए/वाएँ/वाएं/वायें wana/waṯa/waja/ waṯe/wana/waja/ wane/wa/ləwaja/ wae/waẽ/waen/w ajeň	Double Causative Verb Number – Singular Gender – Mescaline
कर/के /ाकर/ने/ाने kar/ke/aakar/ne/a ane	Non Finite Verb

Phrasing or Shallow Parsing: Shallow parsing (also chunking, "light parsing") is an analysis of a sentence which identifies the constituents (noun groups, verbs, verb groups, etc.), but does not specify their internal structure, nor their role in the main sentence. There are various approaches which are used in constructing the Shallow Parser or Creating Phrases is parsing by chunks, Machine Learning approach, Memory Based Shallow Parsing, HMM Based Shallow Parsing, etc.

These approaches have been used extensively in construction of shallow parsers for English and other Latin languages. The approach that is chosen and used for this system is Context Free Grammar based. This approach uses bottom up and left to right parsing for construction of Phrases in a given sentence according to the rules that is described using the CFG Grammar formalism. The approach fails when there is ambiguity in the sentence. Removing ambiguities is not within the current scope of the system. After Shallow Parsing, we can obtain Adjective Phrase, Noun Phrase etc.

Functioning of Grammar Checker System: The user can input a sentence or a group of sentences for checking grammar related errors. System converts it into tokens of the word with the information about its position in paragraph, sentence and gender, number, person, suffix etc.

After that POS Tagging has been done. In this phase morphological clue gives information about POS Category. For example:

सरोज घर जा रहा है।
səroj gʱər ʤa rəɦa ɦɛ।
saroj ghara jaa rahaa hai।

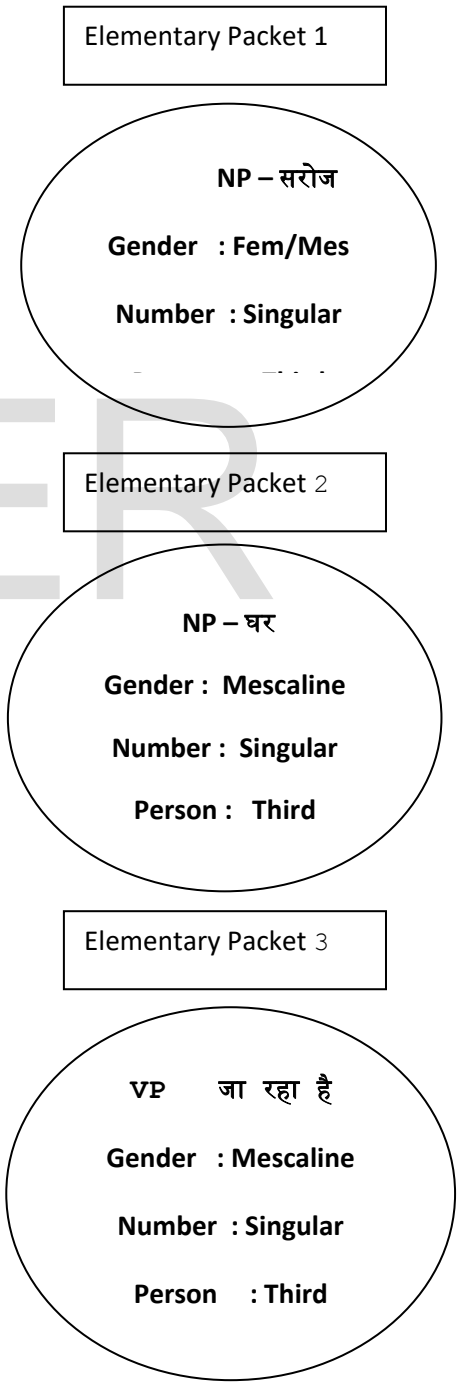
After POS Tagging

सरोज[N] घर[N] जा[VMAIN] रहा [VINF] है [VAUX]

Shallow Parsing

[सरोज N]_NP [घर N]_NP { [जा VMAIN] [रहा VINF] [है VAUX]}_VP

Morphological Analyzer for each elementary packet



There are two methods for finding possible errors in sentence on following level

- Phrase level
- Sentence Level

Phrase level: Error finding method in phrases of sentence is by comparing elementary packets i.e. Phrases of sentences. All associated Elementary Packets have one to one relation in its grammatical feature like Gender, Number and Person etc.

In the above examples, we can see the differences in Elementary Packet 1 and Elementary Packet 3 on gender feature. Now program will give two possibilities of the above sentences.

Possibility I

Suppose that In Elementary Packet 1 is correct the program will change the Gender Number Person (GNP) of Elementary Packet 3 according to GNP features of Elementary Packet 1 thus the output sentence will be:

सीता घर जा रही है।

siṯa gʱər ʤa rəɦi ɦɛ

seetaa ghara jaa rahee hai।

Possibility II

Suppose that Elementary Packet 3 is correct the program will change the Gender Number Person (GNP) of Elementary Packet 1 according to GNP features of Elementary Packet 3 thus the output sentence will be:

सरोज घर जा रहा है।

Sentence Level: Error finding method which we have adopted takes the whole sentence as a unit is called Sentence Level Error Checking Method. The method is required some agreement checking mechanism to find agreements between the Elementary Packets. Therefore we get connected phrases of sentences. This will help in comparing Phrase on the basic of grammatical features and system will ensure whether these features are identical. If the features are identical in both the elementary packets sentences is correct otherwise it is an incorrect sentence.

Conclusion: Design and development of Automatic Grammar Checking facility for Hindi and other Indian languages is possible. The rule based approaches is appropriate for it, because the system will extract information related to grammar at each stage of the system. This will help in deciding the correctness of a sentence. Proposed system has capability for checking in the error in sentence on the basis of Phrases and it associations. But it does not handle the anaphoric relation between the sentences as yet. The system is also capable to find error in various type sentences but it fails to identify the error in some cases. There is every possibility of improving the system so that it covers all types of the sentences.

References :

1. ambati, B. R. (2010). On the role of morphosyntactic features in Hindi dependency parsing. *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages.* Association for Computational Linguistics.

2. Ambati, B. R., Gadde, P., & Jindal, K. (2009). Experiments in Indian Language Dependency Parsing. *Proceedings of ICON09 NLP Tools Contest: Indian Language Dependency Parsing.* Hyderabad.

3. Ambati, B. R., Husain, S., Jain, S., Sharma, D. M., & Sangal, R. (n.d.). Two methods to incorporate local morphosyntactic features in Hindi dependency parsing,. *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of MorphologicallyRich Languages.* Association for Computational Linguistics.

4. Bharati, A. (2009). Simple parser for Indian languages in a dependency framework. *Proceedings of the Third Linguistic Annotation Workshop. Association for Computational Linguistics*.

5. Bharati, A., & Sangal, R. (2009). Parsing Free Word Order Languages in the Paninian Framework. *proceeding ACL '93 Proceeding of the 31st annual meeting on Association for Computational Linguistic*. Association for Computational Linguistic.

6. Bharati, A., Husain, S., Ambati, B. R., Jain, S., Sharma, D. M., & Sangal, R. (2008). Two semantic features make all the difference in parsing accuracy. *In Proceedings of the 6th International Conference on Natural Language Processing.* Pune: ICON-08.

7. bharti, A., Chaitanya, V., & Sangal, R. (2000). *Natural Language Processing : A Paninian Perspective.* New Delhi: Prentice Hall of India Private Limited.

8. Bhattacharya, P. (2012). Natural language Processing: A Perspective from Computation in Presence of Ambiguity. *CSI journal of computing, 1*.

9. Elaine Rich, K. K. (2010). *Artificial Intelligence* . Nodia, India: Tata McGraw-Hill.

10. Gadde, P. (2010). Improving data driven dependency parsing using clausal information, Human Language Technologies . *The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics.* Association for Computational Linguistics.

11. Hopcroft, J. E., & Ullman, J. D. (1996). *Introduction to Automata Theory, Languages and Computation.* New Delhi: Narosa Publishing House.

12. Jain, S. (2013). Exploring Semantic Information in Hindi WordNet for Hindi Dependency Parsing. *The sixth international joint conference on natural language processing.* IJCNLP2013.

13. Jurafsky, D., & Martin, J. H. (2000). *Speech and Language Processing : Introduction to Natural Language Processing, Computational Lingustics, Speech Recognition.* Pearson.

14. Prashanath, M. (2009). Bidirectional Dependency Parser for Hindi, Telugu and Bangla. *Proceedings of ICON09 NLP Tools Contest: Indian Language Dependency Parsing,Hyderabad, India,*.

15. Rahman, M., Das, S., & Sharma, U. (2009). Parsing of part-of-speech tagged Assamese Texts. *IJCSI International Journal of Computer Science Issues, 6*(1).

16. Rich, E., & Knight, K. (1991). *Artificial Intelligence.* McGraw-Hill.

17. Sells, P. (1985). *Lectures on Contemporary Syntactic Theories: An Introduction to Government-binding Theory, Generalized Phrase Structure Grammar, and Lexical-functional Grammar.* Center for the Study of Language and Information, Standford University.

18. van Halteren, H. (1999). *Syntactic Wordclass Tagging.* Springer.

19. कन्हैया सिंह. (2013). *हिंदी भाषा साहित्य और नागरी लिपि*. वाराणसी: विश्वविद्यालय प्रकाशन.

20. कामताप्रसाद गुरु. (2010). *संक्षिप्त हिन्दी व्याकरण.* नई दिल्ली: भारतीय ज्ञानपीठ.

21. तिवारी, ड. भ. (1999). *हिन्दी भाषा कि शब्द संरचना*. नई दिल्ली : साहित्य सहकार.

22. पं. कामताप्रसाद गुरु. (2013). *हिंदी व्याकरण.* नई दिल्ली: तक्षशिला प्रकाशन.

23. पांडेय, म. क. (2008, जनवरी-मार्च). अनुक्रमिक व्याकरण : एक संगणकीय पक्ष. (प. ज. गोपिनाथन, Ed.) *बहुबचन*(18), 112-123.

24. विजय कुमार मल्होत्रा. (2002). *कंप्युटर के भाषिक अनुप्रयोग*. नई दिल्ली : वाणी प्रकाशन.

25. सिंह, ड. स. (2009). *हिंदी का वाकयात्मक व्याकरण.* नई दिल्ली, भारत : प्रभात प्रकाशन.

26. सिंह, स. (2000). *अंग्रेजी-हिंदी अनुवाद व्याकरण*. नई दिल्ली : प्रभात प्रकाशन.