Computational Morphology and Natural Language Parsing for Indian Languages: A Literature Survey

Antony P J
Research Scholar,
Computational Engineering and Networking (CEN), Research Centre,
AMRITA Vishwa Vidyapeetham University,
Coimbatore, India
antonypjohn@gmail.com

Dr K P Soman
Professor and Head,
Computational Engineering and Networking (CEN), Research Centre,
AMRITA Vishwa Vidyapeetham University,
Coimbatore, India
kp_soman}@cb.amrita.edu

Abstract—**Computational Morphology and Natural Language Parsing are the two important as well as essential tasks required for a number of natural language processing application including machine translation. Developing well fledged morphological analyzer and generator (MAG) tools or natural language parsers for highly agglutinative languages is a challenging task. The function of morphological analyzer is to return all the morphemes and their grammatical categories associated with a particular word form. For a given root word and grammatical information, morphological generator will generate the particular word form of that word. On the other hand Parsing is used to understand the syntax and semantics of a natural language sentences confined to the grammar. This literature survey is a ground work to understand the different morphology and parser developments in Indian language. In addition, the paper also deals with various approaches that are used to develop morphological analyzer and generator and natural language parsers tools.**

Keywords- Suffix stripping; Lexicon, Synthesizer; Natural Language Processing; Syntactic Parsing; Grammar Refinement Process; Support Vector Machine; Context Free Grammar.

I. INTRODUCTION

The literature shows that development of morphological analysis and generation as well as natural language parsing work has been successfully done for languages like English, Chinese, Arabic and European languages using various approaches from last few years. Literature shows that there are a very few number of attempts for Indian languages and still these are an ongoing process.

The overall structure of this paper is broadly divided into two sections. The first section deals with the various approaches and different developments in morphological analyzer and generator (MAG) tools for Indian languages. On the other hand the second section gives a literature survey on Indian language natural language parsing.

II. MORPHOLOGICAL APPROACHES AND SURVEY FOR INDIAN LANGUAGES

The morphological structure of an agglutinative language is unique and capturing its complexity in a machine analyzable and generatable form is a challenging job. Analyzing the internal structure of a particular word is an important intermediate stage in many natural language processing applications especially in bilingual and multilingual MT system. A Morphological analyzer is used to analyze the internal structure of the words of a language. On the other hand a morphological generator does exactly the reverse of it i.e. given a root word and grammatical information morphological generator will generate the particular word form of that root word. The role of morphology is very significant in the field of NLP, as seen in applications like MT, question-answering (QA) system, IE, IR, spell checker, lexicography etc. So from a serious computational perspective the creation and availability of a morphological analyzer for a language is important. To build a MAG for a language one has to take care of the morphological peculiarities of that language, specifically in case of machine translation. Some

peculiarities of language such as, the usage of classifiers, excessive presence of vowel harmony etc. make it morphologically complex and thus, a challenge in natural language generation (NLG). The first section of this chapter discuss varies approaches that used for building morphological analyzer and generator tool for Indian languages. The second section gives a brief explanation about different morphological analyzer and generator developments for Indian languages.

A. Morphological Analyzer and Generator Approaches

In general there are several approaches attempted for developing morphological analyzer. In 1983 Kimmo Koskenniemi developed a two-level morphology approach, where he tested this formalism for Finnish language [1]. In this two level representation, the surface level is to describe word form as they occur in written text and the lexical level is to encode lexical units such as stem and suffixes. In 1984 the same formalism was extended in other languages such as Arabic, Dutch, English, French, German, Italian, Japanese, Portuguese, Swedish, Turkish and developed morphological analyzers successfully. In the same time a rule based heuristic analyzer for Finnish nominal and verb forms was developed by Jappinen [2]. In 1996, Beesley [3] developed an Arabic finite state transducer for MA using Xerox finite state transducer (XFST), by reworking extensively on the lexicon and rules in the Kimmo-style. At 2000, Agirve introduced a word–grammar based morphological analyzer using the two-level and a unification- based formalism for a highly agglutinative language called Basque [4]. Similarly using XFST, karine made a Persian MA [5] in 2004 and Wintner came up with a morphological analyzer for Hebrew [6] in 2005. Oflazer Kamel developed a Finite State Machine (FSM) based Turkish morphological analyzer. In 2008, using the syllables and utilizing the surface level clues, the features present in a word are identified for Swahili (or Kiswahili) language by Robert Elwell.

There are many language dependent and independent approaches used for developing morphological analyzer and generator [7]. These approaches can also be classified generally into corpus based, rule based and algorithmic based. The corpus based approach, where a large sized well generated corpus is used for training with a machine learning algorithm. The performance of the system will depends on the feature and size of the corpus. The disadvantage is that corpus creation is a time consuming process. On the other hand, rule based approaches are based on a set of rules and dictionary that contains root and morphemes. In rule based approaches every rule depends on the previous rule. So if one rule fails, it will affect the entire rules that follow it. When a word is given as an input to the morphological analyzer and if the corresponding morphemes are missing in the dictionary then the rule based system fails. Literature shows that there are number of successful morphological analyzer and generator development for languages like English, Chinese, Arabic and European languages using these approaches [8]. Recent development in Indian language NLP shows that many morphological analyzer and generators are created successfully using these approaches. A brief description of most commonly used approaches is as follow:

Corpus Based Approach: In case of corpus based approach, a large sized well generated corpus is required for training. Any machine learning algorithm is used to train the corpus and collects the statistical information and other necessary features from the corpus. The collected information is used as a MAG model. The performance of the system will depends on the feature and size of the corpus. The disadvantage is that corpus creation is a time consuming process. This approach is suitable for languages having well organized corpus.

Paradigm Based Approach: For a particular language, each word category like nouns, verbs, adjectives, adverbs and postpositions will be classified into certain types of paradigms. Based on their morphophonemic behavior, a paradigm based morphological compiler program is used to develop MAG model. In the paradigm approach a linguist or the language expert is asked to provide different tables of word forms covering the words in a language. Based on this information and the feature structure with every word form a MAG can be build. The paradigm based approach is also well suited for highly agglutinative language nature and this or the variant of this scheme has been used widely in NLP. Literature shows that morphological analyzers are developed for almost all Indian languages using paradigm based approach.

Finite State Automata (FSA) Based Approach: Finite state machine or finite state automation FSA (or finite automation) uses regular expressions and is used to accept or reject a string in a given language [9]. In general, an FSA is used to study the behavior of a system composing of state, transitions and actions. When FSA start working, it will be in the initial stage and if the automation is in any one of final state it accept its input and stops working.

Two- Level Morphology Based Approach: In 1983, Kimmo Koskenniemi, a Finnish computer scientist developed a general computational model for word-form recognition and generation called Two- level morphology [9]. This development was one of the major breakthroughs in the field of morphological parsing, which is based on morphotactics and morphophonemics concepts. The advantage of two- level morphology is that the model does not depend on a rule compiler, composition or any other finite-state algorithm. The "two-level" morphological approach consists of two levels called lexical and surface form and a word is represented as a direct, letter-for-letter correspondence between these forms. The Two-level morphology approach is based on the following three ideas:

- Rules are symbol-to-symbol constraints that are applied in parallel, not sequentially like rewrite rules.
- The constraints can refer to the lexical context, to the surface context, or to both contexts at the same time.
- Lexical lookup and morphological analysis are performed in tandem

Finite State Transducers (FST) Based Approach: FST is a modified version of FSA by accepting the principles of a two level morphology. A finite state transducer essentially is a finite state automaton that works on two (or more) tapes. The most common way to think about transducers is as a kind of "translating machine" which works by reading from one tape and writing onto the other. FST's can be used for both analysis and generation (they are bidirectional) and it act as two level morphology. By combining the lexicon, orthographic rules and spelling variations in the FST, we can build a morphological analyzer and generator at once.

Stemmer Based Approach: Stemmer uses a set of rules containing list of stems and replacement rules to stripping of affixes. It is a program oriented approach where developer has to specify all possible affixes with replacement rules. Potter algorithm is one of the most widely used stemmer algorithm and it is freely available. The advantage of stemmer algorithm is that it is very suitable to highly agglutinative languages like Dravidian languages for creating MAG.

Suffix Stripping Based Approach: Highly agglutinative languages such as Dravidian languages, a MAG can be successfully build using suffix stripping approach. The advantage of the Dravidian language is that no prefixes and circumfixes exist for words. Words are usually formed by adding suffixes to the root word serially. This property can be well suited for suffix stripping based MAG. Once the suffix is identified, the stem of the whole word can be obtained by removing that suffix and applying proper orthographic (sandhi) rules. A set of dictionaries like stem dictionary, suffix dictionary and also using morphotactics and sandhi rules, a suffix stripping algorithm successfully implements MAG.

Directed Acrylic Word Graph Based Approach: Directed Acrylic Word Graph (DAWG) is a very efficient data structure that can be used for developing both morphological analyzer and generator. DAWG is language independent and it does not utilize any morphological rules or any other special linguistic information. But DAWG is very suitable for lexicon representation and fast string matching, with a great variety of application. Using this approach, the University of Partas Greece developed MAG for Greek language at first time. There after the method is applied for other languages including Indian languages.

B. MAG for Indian Languages: A Literature Survey

Extensive work has been done already for developing MAG in various Indian languages from last ten to fifteen years. Even though there are many attempts in developing MAG for Indian languages, only few works are publically focused. During the literature survey, I found the following different attempts for developing MAG for Indian language NLP.

MAG for Tamil Language: Literature shows that majority of work in Indian language morphological analyser and generator was done in Tamil languages. The following are the noticeable developments in Tamil MAG. The first five MAG are the recent developments where as the remaining MAG are developed before the year 2007.

i) AMRITA Morph Analyzer and generator for Tamil- A Rule Based Approach (2010): Dr. A.G. Menon, S. Saravanan, R. Loganathan and Dr. K. Soman, Amrita University, Coimbatore, developed a rule based Morphological Analyzer and generator for Tamil using finite state transducer called AMAG [10]. The performance of the system is based on lexicon and orthographic rules from a two level morphological system. The system consists of list of 50000 nouns, around 3000 verbs and a relatively smaller list of adjectives. The proposed AMAG is compared with the existing Tamil morph analyzer and generator called ATCHARAM and proved better performance.

ii) A Novel Algorithm for Tamil Morphological Generator (2010): M.Anand Kumar, V.Dhanalakshmi and Dr. K P Soman ,CEN, Amrita University, Coimbatore developed a morphological generator for Tamil based on suffix stripping algorithm [11]. The system consists of two modules, in which the first module handles the lemma/root part and the second module handles the Morpho-lexical information. The system required the following information: morpho-lexical information file, suffix table, paradigm classification rules and stemming rules. Based on a simple, efficient and language independent algorithm and with a less data, the system efficiently handles compound words, transitive, intransitive and also the proper nouns.

iii) An Improvised Morphological Analyzer cum Generator for Tamil (2010): This work is proposed by Parameswari K, CALTS, university of Hyderabad and deals with the improvised database implemented on Apertium for morphological analysis and generation [12][13]. The improvised MAG uses the Finite State Transducers algorithm for one-pass analysis and generation, and the Word and Paradigm based database. The system performance is measured and compared for its speed and accuracy with the other available Tamil

Morphological analyzers which were developed in CALTS and AU-KBC research Centre, Anna Unicersity. Experiment result showed that the proposed MAG performs better than the other.

iv) A Sequence Labeling Approach to Morphological Analyzer for Tamil (2010): Anand Kumar M, Dhanalakshmi V, Soman K.P and Rajendran S of AMRITA Vishwa Vidyapeetham, Coimbatore, developed morphological analyzer for Tamil language based sequence labeling approach [14]. In the proposed work morphological analyzer problem is redefined as classification problem and solved using machine learning methodology. This is a corpus based approach, where training and testing is performed with support vector machine algorithms. The training corpus consists of 130,000 verb words and 70,000 noun words respectively. The system is tested with 40000 verbs and 30000 nouns taken from Amrita POS Tagged corpus. The performance of the system was also compared with other systems developed using the same corpus and results showed that SVM based approach outperform other.

v) FSA-based morphological generator for Tamil (2010): A finite state automata based morphological generator is developed by Menaka S, Vijay Sundar Ram and Sobha Lalitha Devi [15]. Two separate experiments were conducted for evaluate the system for nouns and verbs using both correct and wrong inputs. The experiment shows that finite-state based morphological generator is well-suited for highly agglutinative and inflectional languages like Tamil.

vi) Rajendran's Morphological Analyzer for Tamil: The first step towards a preparation of morphological analyzer for Tamil was initiated by 'Anusaraka' group of researchers under the guidance of Dr Rajendran [16], Tamil University, Tanjavoor. 'Anusaraka' is machine translation project intended for translation between Indian languages. The developed morphological analyzer for Tamil was used for Translating Tamil language into Hindi at the word level.

vii) Genesan's Morphological Analyzer for Tamil: Ganesan developed a morphological analyzer for Tamil to analyze CIIL corpus. He exploits phonological and morphophonemic rules as well as morphotactic constraints of Tamil in building morphological analyzer. Recently he has built an improved and efficient morphological parser.

viii) Kapilan's Morphological Analyzer for Tamil Verbal Forms: Another attempt was made by Kapilan and he prepared a morphological analyzer for verbal forms in Tamil.

ix) Deivasundaram's Morphological parser: Deivasundarm has prepared a morphological analyzer for Tamil for his Tamil Word Processor. He too makes use of phonological and morphophonemic rules and morphotnatic constraints for developing his parser.

x) AUKBC Morphological Parser for Tamil: AUKBC NLP team under the supervision of Dr Rajendran developed a Morphological parser for Tamil. The API Processor of AUKBC makes use of the finite state machinery like PCKimmo. It parses, but does not generate.

xi) Vishnavi's Morphological Generator for Tamil: Vaishnavi researched for her M.Phil. dissertation on morphological generator for Tamil. The Vaishanvi's morphological generator implements the item and process model of linguistic description. The generator works by the synthesis method of PCKimmo.

xii) Ramasamy's Morphological Generator for Tamil: Ramasamy has prepared a morphological generator for Tamil for his MPhil dissertation.

xiii) Winston Cruz's Parsing and Generation of Tamil Verbs: Winston Cruz makes use of GSmorph method for parsing Tamil verbs. GSmorph too does morphotactics by indexing. The algorithm simply looks up two files to see if the indices match or not. The processor generates as many forms as it parses and uses only two files.

xiv) Vishnavi's Morphological Analyzer for Tamil: Vaishnavi again researched for her Ph.D. dissertation on the preparation of Morphological Analyzer for Tamil. She proposes a hybrid model for Tamil. It finds its theoretical basis in a blend of IA and IP models of morphology. It constitutes an in-built lexicon and involves a decomposition of words in terms of morphemes within the model to realize surface well-formed words-forms. The functioning can be described as defining a transformation depending on the morphemic nature of the word stem. The analysis involves a scanning of the string from the right to left periphery scanning each suffix at a time stripping it, and reconstructing the rest of the word with the aid of phonological and mophophonemic ruels exemplified in each instance. This goes on till the string is exhausted. For the sake of comparison she implements AMPLE and KIMMO models. She also evaluates TAGTAMIL, API Analyzer, and GSMorph. She concludes that Hybrid model is more efficient than the rest of the models.

xv) Dhurai Pandi's Morphological Generator and Parsing Engine for Tamil Verb Forms: It is a full-fledged morphological generator and a parsing engine on verb patterns in modern Tamil.

xvi) RCILTS-T's Morphological analyzer for Tamil: Resource Centre for Indian Language Technological Solutions-Tamil has prepared a morphological analyzer for Tamil. It is named as 'Atcharam'. 'Atcharam' takes a derived word as input and separate into root word and associated morphemes. It uses a dictionary of

20000 root words based on fifteen categories. It has two modules - noun and verb analyzer based on 125 rules. It uses heuristic rules to deal with ambiguities. It can handle verb and noun inflections.

xvii) RCILTS-T's Morphological generator for Tamil: Resource Centre for Indian Language Technological Solutions-Tamil also developed a morphological generator for Tamil. It is named as 'Atchayam'. 'Atchayam' generates words when Tamil morphs are given as input. It has two major modules – noun and verb generators. The noun section handles suffixes like plural markers, oblique form, case markers and postpositions. The verb section takes tense and PNG makers, relative and verbal participle suffixes, and auxiliary verbs. It uses sandhi rules and125 morphological rules. It handles adjectives and adverbs. It has word and sentence generator interfaces.

MAG for Kannada Language: There are five different developments in Kannada language MAG. The first attempt was made by T. N. Vikram and Shalini R Urs in the year 2007 and they developed a morphological analyzer prototype model based on finite state machine. Amrita University, Coimbatore, developed two separate MAG systems for Kannada language. The first Morphological Analyzer developed was based on statistical approach where as the other MAG was based on rule based approach. The R V College of Engineering, Bangalore proposed another morphological analyzer and generator using Trie data structure. Using Network and Process Model, University of Hyderabad developed a MAG system for Kannada language.

i) T. N. Vikram and Shalini R Urs developed a prototype of morphological analyzer for Kannada language (2007) based on Finite State Machine [13]. This is just a prototype and does not handle compound formation morphology and can handle maximum 500 distinct nouns and verbs.

ii) A paradigm based morphological analyzer for Kannada language verbs using the machine learning approach was developed by Antony P J, M Anand Kumar and Dr Soman KP in the year 2010 [17]. The proposed morphological analyzer was designed using sequence labeling approach and training, testing and evaluations were done by support vector method (SVM) algorithms. The system captures the various non-linear relationships and morphological features of Kannada language in a better and simpler way. The performance of the system was evaluated using SVMTeval tool. The system performance is considerably increased by adding more input words to the training corpus whose corresponding output are incorrect during testing and evaluation. From the experiment we found that the performance of our system significantly outperforms and achieves a very competitive accuracy of 96.25% for Kannada verbs.

iii) Ramasamy Veerappan, Antony P J, Saravanan S and Dr Soman KP developed a rule based MAG for Kannada language using finite state transducer (FST) in the year 2011 [18].The proposed MAG is capable of analyzing and generating a list of twenty thousand nouns, around three thousand verbs and a relatively smaller list of adjectives. The uniqueness of the proposed MAG is its capacity to generate and analyze transitive, causative and tense forms apart from the passive constructions, auxiliaries and verbal nouns. This MAG was developed as part of the development of a machine translation system for English to Kannada language.

iv) Kannada Morphological Analyser and Generator Using Trie (2011): Using rule based with paradigm approach, Shambhavi. B. R and Dr. Ramakanth Kumar P of R V College of Engineering, Bangalore proposed a morphological analyzer and generator for Kannada language [8]. They used Trie as a datastructure for the storage of suffixes and root words. The disadvantage of Trie is that it consumes more memory as each node can have at most 'y' children, where 'y' is the alphabet count of the language. As a result it can handle up to maximum 3700 root words and around 88K inflected words.

v) MORPH- A network and process model for Kannada morphological analysis/ generation was developed by K. Narayana Murthy and the reported performance of the system is 60 to 70% on general texts [19]. The advantage of finite state network is that, it captures all the affixes, their ordering and the various combinations permitted in a declarative and bidirectional fashion. Since the same network is used both for analysis and generation, it reduces the overall overhead of the system.

MAG for Malayalam Language: In case Malayalam, there are two different developments in MAG as follow:

i) Malayalam Morphological Analyser and Tamil Morphological Generator for Malayalam - Tamil Machine Translation (2011): Based on suffix stripping and suffix joining approach, using a bilingual dictionary, a Malayalam morphological analyzer and a Tamil morphological generator have been developed by Jisha P.Jayan, Rajeev R R and Dr. S Rajendran [20]. The developed analyzer and generator were used for Malayalam - Tamil machine translation.

ii) Morphological analyzer for Malayalam verbs (2008): Saranya S.K and Dr Soman K P of AMRITA Vishwa Vidyapeetham, Coimbatore developed a prototype morphological analyzer for Malayalam language based on hybrid approach of Paradigm and Suffix Stripping Method [9].

MAG for Hindi Language: Even though there are many attempts in MAG for Hindi language, only one development is available publically. Teena Bajaj proposed a method for extending the range of existing

morphological analyzer system for Hindi language [21]. The work focuses on how strength of existing morph analyzer can be improved by merging it with a semi-supervised approach for learning of Hindi morphology.

MAG for Punjabi Language: Punjabi Morphological Analyzer and Generator (year): There are two different attempts to develop Morphological Analyzer and Generator for Panjabi language [22]. Under 'Anusarka' project, IIIT Hyderabad developed a Punjabi Morph at first time. Later Dr Mandeep Singh, Advanced Centre for Technical Development of Punjabi Language, Punjabi University developed a MAG for Panjabi language.

MAG for Bengali Language: Development of a morphological analyser for Bengali (2009): An open-source morphological analyser for Bengali Language using finite state technology was developed by Abu Zaher Md. Faridee and Francis M. Tyers [23]. This is the first open source attempt in creating a fully-functional morphological analyser and the system is currently in under development stage.

MAG for Assamese, Bengali, Bodo and Oriya Languages: Morphological analyzer using rule based affix stripping approach (2011): The design and development of morphological analyzers for four Indian languages-Assamese, Bengali, Bodo and Oriya was proposed by Mona Parakh and Rajesha N, CIIL Mysore [24]. At present it is an ongoing work based on dictionary based and suffix stripping approach and the performance of the system directly related to the size of the dictionary. The developed prototype model currently can handles inflectional suffixes and work is going to handle derivation as well as prefixation.

III. PARSING APPROACHES AND SURVEY FOR INDIAN LANGUAGES

Parsing of sentences is considered to be an important intermediate stage for semantic analysis in natural language processing (NLP) application such as information retrieval (IR), information extraction (IE) and question answering (QA). The study of structure of sentence is called syntax. It attempts to describe the grammatical order in a particular language in term of rules which in detail explain the underlying structure and a transformational process. Syntax provides rules to put together words to form components of sentences and to put together these components to form meaningful sentences. In natural language processing, syntactic parsing or more formally syntactic analysis is the process of analyzing and determining the structure of a text which is made up of sequence of tokens with respect to a given formal grammar. Because of the substantial ambiguity present in the human language, whose usage is to convey different semantics, it is much difficult to design the features for natural language processing tasks. The main challenge is the inherent complexity of linguistic phenomena that makes it difficult to represent the effective features for the target learning models.

A. Parsing Approaches

A well known parsing approach known as Nivre's parser was successfully implemented in a variety of languages like relatively free-word order language like Turkish, inflectionally rich language like Hindi, fixed word order language like English, and relatively case-less and less inflectional language like Swedish. Another simple approach called 'Context Free Grammar' (CFG) formalism was used in languages like Dutch, Turkish and English to develop parsers. In order to suit the context of Indian languages, a formalism called 'Paninian Grammar Framework' was developed. Collin's and Mc-Donald's parser are the other well known parsing techniques. Generally, natural language parsing can be broadly classified in to three categories: (i) rule based (ii) statistical based and (iii) generalized parsers [25]. All the developed parsers belong to any one of these categories and follow either 'top-down' or 'bottom-up' direction. Statistical and rule based parsing techniques are called 'data-driven' and 'grammar-driven' approaches respectively.

Rule Based Parsers: A rule- based parser uses the hard – coded rules to identify the best parse tree for a given grammar. In a rule based parsing, production rules are applied recursively and as a result overlapping problem may arise. Dynamic programming (DP) technique can be used to solve the overlapping problem efficiently. The cache for sub parse trees in the DP-based parsers is called the 'chart' and consequently the DP-based parsers are called 'chart parsers'. The CYK algorithm developed by Cocke (1970), Kasami (1965), Younger (1967) and Early algorithm developed by Jurafsky and Martin, in 2000 belong to rule based parsers.

Statistical Based Parsers: The main effort in parsing of a sentence is to resolve the ambiguities. It is very hard to write complex rules to resolve such ambiguities. In contrast to the rule based approach, statistical parsing algorithms collect statistical data from correctly parsed sentences, and resolves ambiguity by experience. The advantage of statistical approach is that it covers the whole grammar usage of the language. The performance of the statistical parsers depends on training corpus used to gather statistical information about the grammar of the language. Instead of using rules to find the correct parse tree, statistical parsers select the best parse tree from possible candidates based on the statistical information. Sampson proposed the first statistical based parsing in 1986, using a manually designed language model based on a set of transition networks, and a stimulated annealing decoding search algorithm. CFG and Probabilistic Context Free Grammar (PCFG) based parsers are the examples for statistical parsers.

Generalized parsing: The framework behind both rule based and statistical parsing are similar. Using this advantage, Goodman proposed a general parsing algorithm based on semiring idea. In the year 2005, Melamed

suggested another generalized parsing algorithm which was based on semiring parsing. Melamed generalized algorithm consists of five components such as: grammar, logic, semiring, search strategy and termination condition. As the name suggests, grammar defines terminal and non-terminal symbols, as well as a set of production rules. Logic defines the mechanism of how the parser runs by generating new partial parse trees. The semiring defines how partial parse trees are scored. The search strategy defines the order in which partial parse trees are processed and the termination condition defines when to stop the logic necessarily.

B. Major Parser Developments in Indian Languages: A Literature Survey

Even though natural language parsers play an important role in machine translation, it is still an ongoing process for Indian languages. Comparing with foreign languages, a very little work has been done in the area of natural language processing for Indian languages. This section will give a brief description about various developments contributed towards natural language parsing in Indian languages.

i) Antony P J, Nandini J. Warrier and Dr Soman K P have developed a Penn Treebank based statistical syntactic parsers for Kannada language in 2010 [5]. The well known grammar formalism called Penn Treebank structure was used to create the corpus for proposed statistical syntactic parser. The parsing system was trained using Treebank based corpus which consisted of 1,000 Kannada sentences that was carefully constructed. The developed corpus has been already annotated with correct segmentation and Part-Of-Speech (POS) information. The developers used their own SVM based POS tagger generator for assigning proper tags to each and every word in the training and test sentences. The proposed syntactic parser was implemented using supervised machine learning and probabilistic context free grammars (PCFG) approaches. Training, testing and evaluation were done by support vector method (SVM) algorithms. Experiment shows that the performance of the proposed system is significantly good and has very competitive accuracy.

ii) In the year 2009, B.M. Sagar, Shobha G and Ramakanth Kumar P proposed a way of producing context free grammar for the Noun Phrase and Verb Phrase agreement in Kannada Sentences [6]. In this approach, a recursive descent parser is used to parse the CFG. The system works in two stages: First, it generates the context free grammar of the sentence. In the second stage, a recursive descent parser called Recursive Descent Parser of Natural Language Tool Kit (NLTK) was used to test the grammar. As a summary, it is a grammar checking system such that for a given sentence parser says whether the sentence is syntactically correct or wrong depending upon the Noun and Verb agreement. They have tested the system using around 200 sample sentences and obtained encouraging results.

iii) Natural Language constructs for Venpa class of Tamil Poetry using Context Free Grammar was implemented by Bala Sundara Raman L, Ishwar S, and Sanjeeth Kumar Ravindranath in 2003 [7]. They used Push Down Automata parser to parse the CFG in the proposed system.

iv) A rule based grammar checking mechanism for Hindi sentences was developed by Singh and D.K. Lobiyal in 1993 [8]. The system is suitable for all types of sentences with compound, conjunct or complex verb phrases. In the proposed model, verb and suffixes have been divided into finer subcategories to simplify the process of associating semantics with syntax. Using Lexical Functional Grammar formalism, the base grammar rules are being augmented with functional equations. This technique is used to bridge the gap between syntax and semantics of a sentence. They have developed a parser for the grammar. The grammar rules are assigned priority in a manner that the most frequently applicable rule gets higher priority than the less frequently applicable rule. The system works in such a way that, the grammar rules get fired on priority basis to make the parsing efficient.

v) Selvam M, Natarajan. A M, and Thangarajan R proposed a statistical parsing of Tamil sentences using phrase structure hybrid language model in the year 2008 [9]. They have built a statistical language model based on Trigram for Tamil language with medium of 5000 words. In the experiment they showed that statistical parsing gives better performance through tri-gram probabilities and large vocabulary size. In order to overcome some disadvantages like focus on semantics rather than syntax, lack of support in free ordering of words and long term relationship of the system, a structural component is to be incorporated. The developed hybrid language model is based on a part of speech tagset for Tamil language with more than 500 tags. The developed phrase structured Treebank was based on 326 Tamil sentences which covers more than 5000 words. The phrase structured Treebank was trained using immediate head parsing technique. Two test cases with 120 and 40 sentences have been selected from trained set and test set respectively. They reported that, the performance of the system is better than the grammar model.

vi) Akshar Bharati and Rajeev Sangal described a grammar formalism called the 'Paninian Grammar Framework' that has been successfully applied to all free word Indian languages [10]. They have described a constraint based parser for the framework. Paninian framework uses the notion of karaka relations between verbs and nouns in a sentence. They showed that the Paninian framework applied to modern Indian languages will give an elegant account of the relation between vibhakti and karaka roles and that the mapping is elegant and compact.

vii) B.M. Sagar, Shobha G and Ramakanth Kumar P proposed a Context Free Grammar (CFG) Analysis for simple Kannada sentences in 2010 [11]. They have explained the writing of Context Free Grammar (CFG) for a simple Kannada sentence with two sets of examples. In the proposed system, a grammar is parsed with Top-Down and Bottom-Up parsers and they found that a Top-Down parser is more suitable to parse the given grammatical production.

viii) A dependency parser system for Bengali language was developed by Aniruddha Ghosh, Pinaki Bhaskar, Amitava Das and Sivaji Bandyopadhyay in 2009 [12]. They have performed two separate runs for Bengali. A statistical CRF based model followed by a rule-based post-processing technique has been used. They have used ICON 2009 datasets for training the system. They have trained the probabilistic sequence model with the morphological features like root word, POS-tag, chunk tag, vibhakti and dependency relation from the training set data. The output of the baseline CRF based system is filtered by a rule-based post-processing module by using the output obtained through the rule based dependency parser. The system demonstrated an unlabeled attachment score (UAS) of 74.09%, labeled attachment score (LAS) of 53.90% and labeled accuracy score (LS) of 61.71% respectively.

ix) Sengupta,P.and B.Chaudhuri proposed a delayed syntactic-encoding-based LFG parsing strategy for an Indian Language- Bangla, in 1997 [13]. This was just an attempt in parsing of Bengali language based on the detection and formation of the proper rule set to identify characteristics of inter-chunk relations. They have tested the system on a sample of about 250 simple and complex sentences picked from newspaper clippings. Results show that, even though phrasal orderings were quite random, almost all simple sentences in active voice were correctly parsed.

x) Parsing of Indian Languages using the freely available Malt- Parser system was developed by Joakim Nivre in 2009 [14]. He developed transition-based dependency parsers using Malt- Parser system for three Indian languages like Bangla, Hindi and Telugu. With the Malt- Parsing technique he showed that, parsing can be performed in linear time for projective dependency trees and quadratic time for arbitrary trees. A small test set of 150 sentences was used to analyse the performance of the system. The performance of the system was slightly better for Bangla and Hindi languages but for Telugu it was lower than the baseline results.

xi) Hiring world's leading dependency parsers to plant Indian trees, a voting parser was proposed by Daniel Zeman in 2009 [15] called Maximum Spanning Malt. The system consists of three existing, freely available dependency parsers, two of which (MST and Malt) have been known to produce state-of-the-art structures on data sets for other languages. Various settings of the parsers were explored in order to adjust them for the three Indian languages like Hindi, Bengali and Telugu, and a voting approach was used to combine them into a super parser. He showed that 'case' and 'vibhakti' are important features for parsing Hindi while their usability in Bangla and Telugu is limited by data sparseness. Based on these features, he developed best combined parsers for these languages.

xii) A constraint based Dependency parsing has been attempted and applied to a free-word order language Bangla by Sankar, Arnab Dhar and Utpal Garain in 2009 [16]. They have used a structure simplification and demand satisfaction approach to dependency parsing in Bangla language. A well known and very effective grammar formalism for free word order language called Paninian Grammatical model was used for this purpose. The main idea behind this approach was to simplify complex and compound sentential structures first, then to parse the simple structures so obtained by satisfying the 'Karaka' demands of the Demand Groups (Verb Groups) and to rejoin such parsed structures with appropriate links and Karaka labels. A Treebank of 1000 annotated sentences was used for training the system. The performance of the system was evaluated with 150 sentences and achieves accuracies of 90.32%, 79.81%, and 81.27% for unlabeled attachments, labelled attachments and label scores, respectively.

xiii) Bharat Ram Ambati, Phani Gadde and Karan Jindal explored two data-driven parsers called Malt and MST on three Indian languages namely Hindi, Telugu and Bangla in 2009 [17]. In their experiment, they merged both the training and development data and did 5-fold cross-validation for tuning the parsers. They also extracted best settings from the cross validation experiments and these settings are applied on the test data of the contest. Finally they evaluated the individual and average results on both coarse-grained and fine-grained tagset for all the three languages. They observed that for all the languages Malt performed better over MST+maxent. They also modified the implementation of MST to handle vibhakti and TAM markers for labelling. They reported that, the average of best unlabeled attachment, labelled attachment and labelled accuracies are 88.43%, 71.71% and 73.81% respectively.

xiv) A hybrid approach for parsing Bengali sentences was proposed by Sanjay Chatterji, Praveen Sonare, Sudeshna Sarkar and Devshri Roy in 2009 [18]. The system was based on data driven dependency parser. In order to improve the performance of the system, some handcrafted rules are identified based on the error patterns on the output of the baseline system.

xv) A constraint based Hindi dependency parser was developed by Meher Vijay Yeleti and Kalyan Deepak in 2009 [19]. In the proposed system a grammar driven approach was complemented by a controlled statistical

strategy to achieve high performance and robustness. The proposed system uses two stage constraint based hybrid approach to dependency parsing. They defined two stages and this division leads to selective identification and resolution of specific dependency relations at the two stages. They also used hard constraints and soft constraints to build an efficient and robust hybrid parser. From the experiment they found out that the best labelled and unlabeled attachment accuracies for Hindi are 62.20% and 85.55% respectively.

xvi) Prashanth Mannem proposed a bidirectional dependency parser for Hindi, Telugu and Bangla languages in 2009 [20]. The developed parser uses a bidirectional parsing algorithm with two operations projection and non-projection to build the dependency tree. The performance of the proposed parser was evaluated based on the test data sentences. He reported that the system achieves a labelled attachment score of 71.63%, 59.86% and 67.74% for Hindi, Telugu and Bangla respectively on the treebank with fine-grained dependency labels. Based on the coarse-grained labels the dependency parser achieved 76.90%, 70.34% and 65.01% accuracies respectively.

xvii) Pradeep Kumar Das proposed a generative approach to the computation of basic verbal-strings in Hindi in 2008 [21]. He described a way to examine the possibility of developing a computational parser for verb morphology in Hindi that would generate correct verbal stems for different kinds of tense and aspects.

xviii) A parsing criteria for Assamese text was described by Navanath Saharia, Utpal Sharma and Jugal Kalita in 2011 [22]. They described the practical analysis of Assamese sentences from a computational perspective rather than from linguistics perspective. This approach can be used to parse the simple sentences with multiple noun, adjective and adverb clauses.

xix) An attempt to study the semantic relation of Causality or Cause-Effect was proposed by Sobha Lalitha Devi and Menaka S in 2011 [23]. They also described how semantic relation of Causality is marked in Tamil, how the causal markers in Tamil manifest in texts, their syntactic and semantic properties and how this information can be represented so as to handle causal information and reasoning.

xx) Akshar Bharati, Mridul Gupta, Vineet Yadav, Karthik Gali and Dipti Misra Sharma proposed a simple parser for Indian Languages in a dependency framework in 2009 [24]. They described a parser which uses a grammar driven approach to annotate dependency relations in both inter and intra chunk at an intermediary level. They described a grammar oriented model that makes use of linguistic features to identify relations. The proposed parser was modelled based on Paninian grammatical approach which provides a dependency grammar framework. They also compared the proposed parser performance against the previous similar attempts and reported its efficiency.

xxi) An approach to expedite the process of manual annotation of a Hindi dependency Treebank was described by Gupta, Vineet Yadav, Samar Husain and Dipti Misra Sharma [25]. They proposed a way by which consistency among a set of manual annotators could be improved. They have also showed that their setup can useful for evaluating, when an inexperienced annotator is ready to start participating in the production of the treebank. The performance of the system was evaluated on sample sets of data.

xxii) An unlabeled dependency parsing on graph based method for building multilingual weakly supervised dependency parsers for Hindi language was proposed by Jagadeesh Gorla, Anil Kumar Singh, Rajeev Sangal, Karthik Gali, Samar Husain, and Sriram Venkatapathy [5]. The system consists of two steps where the first step involves marking the chunks and the chunk heads of a given sentence and then identifying the intra-chunk dependency relations. The second step involves learning to identify the inter-chunk dependency relations. They reported that the system achieved a precision of 80.83% for sentences of size less than 10 words and 66.71% overall. They concluded that the result obtained was significantly better than the baseline in which random initialization is used.

IV. CONCLUSION

In this paper work, we have presented a survey on different developments of morphological analyzer and generator as well as natural language parsers for Indian languages. Additionally I tried to give a brief idea about the existing approaches that have been used to develop morphological analyzer and generator as well as natural language parsers for Indian languages. From the survey we have found that almost all approaches were applied in different morphological analyzer and generator for Indian languages. On the other hand, the literature reveals that almost all existing Indian languages parsers are based on statistical and hybrid approach. We also noted that, the main effort and challenge behind each and every development is to design the system by considering the agglutinative and morphological rich features of language.

ACKNOWLEDGMENT

We acknowledge our sincere gratitude to Mr. Benjamin Peter (Assistant Professor, MBA Dept, St.Joseph Engineering College, Mangalore, India) and Mr. Rakesh Naik (Assistant Professor, MBA Dept, St.Joseph Engineering College, Mangalore, India) for their valuable support regarding proof reading and correction of this survey paper.

REFERENCES

[1] Uma Parameshwari Rao G, Parameshwari K: CALTS, University of Hyderabad, 'On the description of morphological data for morphological analyzers and generators: A case of Telugu, Tamil and Kannada'.

[2] Harri Jappinen, 'Knowledge engineering approach to morphological analysis', first conference on European chapter of the Association for Computational Linguistics.

[3] Beesley, K. and L. Karttunen. 'Finite State Morphology'. Stanford, CA: CSLI Publications, 2003.

[4] Aduriz 1., Agirre E., 'A word-grammar based morphologieal analyzer for agglutinative languages', University of lhe Basqtlo Cotlnlry, Basque Country.

[5] Karine Megerdoomian 'Finite-State Morphological Analysis of Persian', Inxight Software, Inc, University of California, San Diego.

[6] Shuly Winter, 'Hebrew Computational Linguistics: Past and Future', Artificial Intelligence Review 21: 113–138, 2004, Kluwer Academic Publishers.

[7] Choudhary and Narayan Kumar, 'Developing a Computational Framework for the Verb Morphology of Great Andamanese', Centre for Linguistics, JNU, 2006.

[8] Shambhavi. B. R, Dr. Ramakanth Kumar P, Srividya K, Jyothi B J, Spoorti Kundargi, and Varsha Shastri G, " Kannada Morphological Analyser and Generator Using Trie", International Journal of Computer Science and Network Security (IJCSNS), VOL.11 No.1, January 2011.

[9] Saranya S.K, 'Morpphological Analyzer for Malayalam Verbs'- A Project Report, www.pdfcreat.com/computer-malayalam-0.html. 2008.

[10] Dr. A.G. Menon, S. Saravanan, R. Loganathan and Dr. K. Soman,, "Amrita Morph Analyzer and Generator for Tamil: A Rule Based Approach", Tamil International Conference, 2010, Coimbatore, India.

[11] M.Anand Kumar, V.Dhanalakshmi and Dr. K P Soman, "A Novel Algorithm for Tamil Morphological Generator", ICON 2010 IIT-Kharagpur, 2010.

[12] Parameswari K, "An Improvized Morphological Analyzer cum Generator for Tamil: A case of implementing the open source platform APERTIUM", Knowledge Sharing Event 1 – CIIL, Mysore, March 2010.

[13] Parameshwari K, "An Implementation of APERTIUM Morphological Analyzer and Generator for Tamil", Language in India www.languageinindia.c o m 1 1 : 5 M a y 2 0 1 1, Special Volume: Problems of Parsing in Indian Languages, 2011.

[14] Anand Kumar M, Dhanalakshmi V, Soman K.P and Rajendran S, "A Sequence Labeling Approach to Morphological Analyzer for Tamil Language", (IJCSE) International Journal on Computer Science and Engineering Vol. 02, No. 06, 2010, 1944-195.

[15] Menaka S, Vijay Sundar Ram and Sobha Lalitha Devi, 'Morphological Generator for Tamil', Morphological Analysers and Generators", (ed.) Mona Parakh, LDC-IL, Mysore, pp. 82 –96, 2010.

[16] S. Rajendran, "Parsing in Tamil", LANGUAGE IN INDIA www.languageinindia.com Vol 6 : 8 August, 2006.

[17] Antony P J, Anand Kumar M and Soman K P: "Paradigm Based Morphological Analyzer for Kannada Language Using Machine Learning Approach.", International journal on-Advances in Computer Science and Technology (ACST), ISSN 0973-6107, Vol 3 No. 4, 2010, pp. 457–481.

[18] Ramasamy Veerappan, Antony P J, Saravanan S and Soman K P: "Rule based POS tagger for Kannada language using Finite State Transducer ", International journal on Computer Application (IJCA), ISBN: 978-93-80746-92-0, 2011.

[19] K. Narayana Murthy, "A Kannada morphological analyzer and generator using Network and Process Model", Resource Centre For Indian Language Technology Solutions – Telugu University of Hyderabad, http://www.tdil.mit.gov.in/Telugu-UOHJuly03.pdf.

[20] Jisha P.Jayan, Rajeev R R and Dr. S Rajendran, "Morphological Analyser and Morphological Generator for Malayalam - Tamil Machine Translation", International Journal of Computer Applications (0975 – 8887, Volume 13– No.8, January 2011.

[21] Teena Bajaj, "Rule Based Semi-Supervised Morphological Analyzer for Extending the Range of Existing System", Thesis submitted in partial fulfillment of the requirements for the award of degree of Master of Engineering in Software Engineering, 2008.

[22] Dr Mandeep Singh, "A Punjabi Morphological Analyzer and Generator", Advanced Centre for Technical Development of Punjabi Language, Literature and Culture, Punjabi University.

[23] Abu Zaher Md. Faridee and Francis M. Tyers, "Development of a morphological analyser for Bengali", Proceedings of the First International Workshop on Free/Open-Source Rule-Based Machine Translation, p. 43-50, Alacant, Spain, November 2009.

[24] Mona Parakh and Rajesha N, 'Developing Morphological Analyzer for Four Indian Languages Using A Rule Based Affix Stripping Approach', Linguistic Data Consortium for Indian Languages, CIIL, Mysore, 2011.

[25] Chinese-English Statistical Machine Translation by Parsing by Yue Zhang Mansfield College, Computing Laboratory, University of Oxford, 2006., www.cl.cam.ac.uk/~yz360/ mscthesis.pd.

[26] Reut Tsarfaty Yoav Goldberg, "Word-Based or Morpheme-Based? Annotation Strategies for Modern Hebrew Clitics".

[27] Abhishek Arun, (2004), "Statistical Parsing of the French Treebank", A thesis for Master of Science, Cognitive Science and Natural Language, School of Informatics, University of Edinburgh.

[28] Ayesha Binte Mosaddeque & Nafid Haque, (2004), "Context-Free Grammar for Bangla", Bangla, Dhaka, Bangladesh.

[29] Antony P J, Nandini. J. Warrier and Soman K P: "Penn Treebank-Based Syntactic Parsers for South Dravidian Languages using a Machine Learning Approach", International journal on Computer Application (IJCA), No. 08, ISBN: 978-93-80746-92-0, 2010.

[30] Solving the Noun Phrase and Verb Phrase Agreement in Kannada Sentences by B.M. Sagar, Shobha G and Ramakanth Kumar P. International Journal of Computer Theory and Engineering, Vol. 1, No. 3, August, 2009, 1793-8201.

[31] Context Free Grammar for Natural Language Constructs – An Implementation for Venpa class of Tamil Poetry by Bala sundara Raman L, Ishwar S, Sanjeeth Kumar Ravindranath, , Tamil Internet 2003, Chennai, India. Implemented Natural Language constructs for Venpa class of Tamil Poetry using Context Free Grammar. Push Down Automata is the parser used to parse the CFG.

[32] A Computational Grammar for Hindi Verb Phrase by Singh and D.K. Lobiyal, (1994), Proceedings of International Conference on Expert Systems for Development, 1994., ieeexplore.ieee.org/iel2/973/ 7458/00302273.pdf.

[33] Structural Parsing of Natural Language Text in Tamil Using Phrase Structure Hybrid Language Model by Selvam M, Natarajan. A M, and Thangarajan R. World Academy of Science, Engineering and Technology 39, 2008.

[34] Parsing Free Word Order Languages in the Paninian Framework by Akshar Bharati and Rajeev Sangal, www.ldc.upenn.edu/acl/P/P93/P93-1015.pdf.

[35] Context Free Grammar (CFG) Analysis for simple Kannada sentences by B.M. Sagar, Shobha G and Ramakanth Kumar P. Special Issue of IJCCT Vol.1 Issue 2, 3, 4; 2010 for International Conference [ACCTA-2010], 3-5 August 2010.

[36] Dependency Parser for Bengali: the JU System at ICON 2009 by Aniruddha Ghosh, Pinaki Bhaskar, Amitava Das and Sivaji Bandyopadhyay. Proceedings of ICON09 NLP Tools Contest: Indian Language Dependency Parsing, Hyderabad, India, 2009.

[37] A Delayed Syntactic-Encoding-based LFG Parsing Strategy for an Indian Language- Bangla by Sengupta,P.and B.Chaudhuri. Association for Computational linguistics 1997, www.sivajibandyopadhyay.com /pinaki/papers/ICON-2009_Parser_JU.pdf.

[38] Parsing Indian Languages with MaltParser by Joakim Nivre, Proceedings of ICON09 NLP Tools Contest: Indian Language Dependency Parsing, Hyderabad, India, 2009.

[39] Maximum Spanning Malt: Hiring World's Leading Dependency Parsers to Plant Indian Trees by Daniel Zeman. Proceedings of ICON09 NLP Tools Contest: Indian Language Dependency Parsing, Hyderabad, India, 2009.

[40] Structure Simplification and Demand Satisfaction Approach to Dependency Parsing in Bangla by Sankar, Arnab Dhar and Utpal Garain, Proceedings of ICON09 NLP Tools Contest: Indian Language Dependency Parsing, Hyderabad, India, 2009.

[41] Experiments in Indian Language Dependency Parsing Bharat Ram Ambati, Phani Gadde and Karan Jindal, Proceedings of ICON09 NLP Tools Contest: Indian Language Dependency Parsing, Hyderabad, India, 2009.

[42] Grammar Driven Rules for Hybrid Bengali Dependency Parsing by Sanjay Chatterji, Praveen Sonare, Sudeshna Sarkar and Devshri Roy, Proceedings of ICON09 NLP Tools Contest: Indian Language Dependency Parsing, Hyderabad, India, 2009.

[43] Constraint based Hindi dependency parsing by Meher Vijay Yeleti and Kalyan Deepak, Language Technologies Research Centre, Proceedings of ICON09 NLP Tools Contest: Indian Language Dependency Parsing, Hyderabad, India, 2009.

[44] Bidirectional Dependency Parser for Hindi, Telugu and Bangla by Prashanth Mannem, Language Technologies Research Center, International Institute of Information Technology, Proceedings of ICON09 NLP Tools Contest: Indian Language Dependency Parsing, Hyderabad, India, 2009.

[45] The Generative approach to the computation of Basic verbal-strings in Hindi by Pradeep Kumar Das, The paper was presented in the conference "Global Association of Indo-ASEAN Studies" held at Pusan University of Foreign Studies, Pusan, South Korea, 14-11-2008.

[46] A First Step Towards Parsing of Assamese Text by Navanath Saharia, Utpal Sharma and Jugal Kalita, 2 0 1 1 Special Volume: Problems of Parsing in Indian Languages.

[47] Semantic Representation of Causality by Sobha Lalitha Devi and Menaka S, May 2011, Special Volume: Problems of Parsing in Indian Languages.

[48] Simple Parser for Indian Languages in a Dependency Framework by Akshar Bharati, Mridul Gupta, Vineet Yadav, Karthik Gali, Dipti Misra Sharma in Proceedings of the Third Linguistic Annotation Workshop (LAWIII), SIGANN, 47th ACL - 4th IJCNLP, Singapore (IJCNLP-2009).

[49] Partial Parsing as a Method to Expedite Dependency Annotation of a Hindi Treebank, Mridul Gupta, Vineet Yadav, Samar Husain and Dipti Misra Sharma Language Technologies Research Center, International Institute of Information Technology, Hyderabad. Language Resources, Technologies and Evaluation Conference-2010(LREC'2010).www.lrec-conf.org/proceedings/lrec2010/summaries /739.html.