This section briefly describes existing parsing techniques which are useful to design machine translation systems. In here, parsing techniques are categorized from top-down and bottom-up.

**Top-down parsers**
The Top-down parsers are analyzed the input source left to right and searching for parse trees using a top-down expansion. Some parsers developed through the top-down parsing including Recursive descent parser, LL parser, Early Parser, X-SAIGA parser, etc. These parsers have demonstrated their properties and capabilities to syntax analysis.

The Recursive descent parser is the straightforward forms of top-down parsing. This type of parser attempts to verify the syntax of the sentence is corrected as it is read from left to right. The LL Parser is also used top-down parsing and parses the input from left to right and constructs a Leftmost derivation of the sentence for context-free languages. The ANTLR is the popular LL parser, especially for compilers. The LL(k) parser uses the above techniques to parse the sentences without backtracking. The Early parsers are especially suitable for ambiguous grammars and use for parsing the computational linguistics. Many of these parsers are already implemented through the C, Java, Perl and Python languages. The X-Saiga parsers are developed under the X-Saiga project to create algorithms and implementations. X-Saiga parsers enable the construction of language processors such as recognizers, parsers, interpreters, translators, etc.

**Bottom-up parsers**
The bottom-up parser attempts to identify the most fundamental units first. Then it attempts to build trees upwards the start. These parsers are mainly used to analyze both natural languages and computer languages. Some parsers are already developed using bottom-up approach including Operator Precedence Parsers, LR parsers CYK parsers, etc.

The operator precedence parser is a bottom-up parser that interprets an operator-precedence grammar. These types of parsers also use a precedence table to determine the next action. The LR Parser uses bottom-up parsing technique and parses the input from Left to the right. Then LR parser constructs a rightmost derivation of the sentence. The CYK Parser is another bottom-up parser that uses the bottom-up parsing technique. The CYK parsers operate on context-free grammars given in Chomsky normal form (CNF).

**TOOLS FOR PARSER DEVELOPMENT**
Parsers are developed through the several computer languages especially Prolog, Java and Python. This section gives a brief description of existing parser development tools that can be used to develop parsers for Natural Language Processing. There are some tools available to
develop Natural Language Processing applications including NLTK, OpenNLP, ProNTO and JavaCC. Further, Prolog (especially in Sri-prolog) gives a set of facilities/libraries for the Natural Language Processing.

Natural Language Toolkit (NLTK) is a toolkit to develop Natural Language Processing applications through the Python programs. This library provides facilities for the area of tokenization, stemming, tagging, parsing, and semantic reasoning. NLTK is free, open source, community-driven project and available for Windows, Mac OS X, and Linux platform. The current version of the NLTK is 3.0.1 which was released in January 2015.

OpenNLP CCG Library is an another open source Natural Language Processing library that is written in Java, which provides parsing and realization services based on Combinatory Categorical Grammar (CCG) formalism. This library provides machine learning based toolkit for the processing of natural language text including tokenization, sentence segmentation, part-of-speech tagging, and named entity extraction, chunking, and parsing.

ProNTO Prolog Natural Language Tools are usable software package which was developed to make Natural Language Processing applications through the SWI-Prolog.

Java Compiler (JavaCC) is a popular Parser Generator that reads a grammar specification and converts it to a Java program. This tool also provides other standard capabilities such as tree building, actions and debugging. JavaCC works with any Java VM version 1.2 or greater.

Further, Prolog-based parsers are stranded parsers that are developed through the Natural Language Processing techniques available in Prolog.

EXISTING PARSERS
This section briefly describes some existing parsers for Natural Language Processing including Stanford parser, Link Grammar Parser, ANTLER, etc.

**Stanford parser**
Stanford parser is a high-performance probabilistic parser that syntactically analyzes a given sentence and provides Stanford Dependencies (SD) output as well as phrase structure trees. The Stanford typed dependencies representation was designed to deliver a simple description of the grammatical relationships in a sentence that can be understood easily and efficiently used by people without linguistic expertise. This SD is triplets such as the name of the relation, governor and dependent. The Standford parser improves its performances through the Shift-reduce constituency parser and Neural-network dependency parser. The shift-reduce parser is much faster constituent parser with competitive accuracy. The Neural-network dependency parser is also high-performance dependency parser powered by a neural network. In addition to the English parser, the parser has been adapted to work with other languages such as Chinese, German and Arabic. However, these statistical parsers still make some mistakes, but commonly work rather well.
The Link Grammar Parser
The Link Grammar Parser is a syntactic parser of English which is based on link grammar. Link grammar is a theory of syntax which was introduced by Davy and Daniel. This grammar is a formal grammatical system, shows how English grammar can be encoded in such a system, and gives algorithms for efficiently parsing with a link grammar. The Link Grammar parser was written in generic C code and runs on any platform with a C compiler. The parser comes with some features such as it is able to skip over portions of the sentence that it cannot understand, and assign some structure to the rest of the sentence. Therefore, this parser becomes a robust Parser. The Link Grammar Parser has coverage of a wide variety of syntactic constructions including many rare and idiomatic ones and has a dictionary of about 60000-word forms. At present link grammar parser is now being maintained under the auspices of the AbiWord project.

The Enju Parser
Enju is a syntactic parser for English which was developed by the Laboratory, Department of Computer Science, The University of Tokyo. Enju Parser works through the wide-coverage probabilistic HPSG grammar and an efficient parsing algorithm. The Enju parser can effectively analyze syntactic/semantic structures of English sentences and provide a user with phrase structures and predicate-argument structures. Compare with others Enju comes with several features including accurate deep analysis, High speed; Output parse results in an XML format, etc. Fuji also supports for the Windows, Linux, and Mac OS X systems as well.

Semantic role labeler
Anders and et.al have developed a high-performance Semantic parser for English name semantic role labeler. This English parser consists of two components namely converter and the parser. A converter transformed constituent parse trees into dependency graphs and the parser of the system constructed semantic dependencies. A present system capable of handle only single sentence at once.

Logon Parser
High-efficiency English Parser has been developed for the Norwegian to English Machine translation project (LOGON) using LinGO English Resource Grammar (ERG). The LinGO English Resource Grammar is a broad-coverage, linguistically precise HPSG-based grammar of English. The LOGON project integrates in-depth grammatical analysis of Norwegian with precise logical-form meaning representations and full, grammar-based generation for English. Sample Parser is also available on the web.
**EP4IR Parser**

Computer Science Department of the Radboud University of Nijmegen has been developed an English parser under the AGFL project. The goal of the project is the development of technology and resources for Natural Language Processing available in the public domain. The parser has also developed through the AGFL grammar and includes a lexicon system which can cope efficiently with the large lexica needed in real-life NLP applications. AGFL system is one of the parser-generator for linguistic applications which was made available under the GNU General Public License.

**Shallow Parser**

Daelemans and other have developed an English parser through the Memory-Based Shallow Parsing. Memory-Based Shallow Parsing system consists of Part of Speech tagging, Chunking, Lemmatizing, Relation Finding and Semantic tagging.

**Indian Region parsers**

There are some parsers already developed for the Indian regain to develop Natural Language Processing applications, especially for Machine Translation.

Hamburg and Srivastav have developed Hindi parser by using CYK algorithms. Hindi CKY Parser uses a handful of grammar rules that define the whole Hindi language. They have used some rules to cover the assertive sentences of Hindi languages and based on these rules they apply parsing and build the tree structure of the sentences.

Fernando and Weerasinghe have developed a morphological Sinhala parser that is capable of analyzing and generating Sinhala verbs. The parser consists of a lexicon of more than 400 verb stems and handles 45 inflectional rules for each stem. The system has been modeled through the framework of two-level morphology model which are available on Xerox Finite state morphology tools.

English parser has already developed for the English to Sinhala Machine Translation system (BEES). The English parser has been implemented to analyze the given English sentence. This parser has been implemented through the SWI-Prolog. To analyze the given English sentence, it uses original English sentence and the result of the English morphological analysis. The Parser handles only simple sentence and provides facilities for translating English sentence into Sinhala. This parser gets support from English morphological analyzer with lexical dictionaries to parse the English sentence.