**IJCTA,** 9(10), 2016, pp. 4409-4421


# Open Source NLG Systems : A Survey with A Vision to Design a True NLG System

***Sandhya Singh **Hemant Darbari ***Krishnanjan Bhattacharjee ****Seema Verma**

***Abstract :*** A Natural Language Generation (NLG) system is computer software that can generate natural language text like humans with the help of language knowledge and domain knowledge of the context. The ultimate aim of researchers from the field of Artificial Intelligence (AI) is to generate text with human kind of intelligence and cognition ability. Researchers have successfully designed and experimented with domain specific NLG systems. This paper presents a systematic study of the NLG tools designed and implemented so far with the idea to understand their scope, architecture and performance. Also analyze the link needed towards the designing of true NLG system.

***Keywords :*** Natural Language Generation(NLG), Natural Language Understanding (NLU), Natural Language Processing (NLP), Artificial Intelligence(AI), Cognition.

## 1. INTRODUCTION

The area of Artificial intelligence (AI) with the aim to design machines, which are capable of simulating humans, has seen many path breaking results. With IBM Watson like supercomputer, researchers have already shown the possibilities of combining AI with some level of cognition through sophisticated analysis. Techniques like machine learning, reinforcement learning, and dynamic programming are being used to simulate human kind of intelligence in various areas.

Natural Language Processing (NLP) is one area of AI where researchers are experimenting to process the natural language with human ease through machines. The challenge increases with the number of languages used across the world, which ranges from 3000 to 8000 different languages. According to Ethnologue[1], there are around 147 language families in the world. These numbers are indicative enough to realize the challenges in language processing.

Many tasks of NLP like Part of Speech (POS), Named Entity Recognition (NER), Parsing, word segmentation etc. have been solved successfully for many languages. Other tasks like machine translation, which requires expertise in grammar, syntax (sentence structure), semantics (meanings), etc., in the source and target languages have also been solved to a great extent in NLP. System like Google translate has demonstrated the ability of translating around 90 language pairs with decent accuracy. Still challenges like NLG which produce natural language expressions from non-linguistic input is far from reality. The area of NLG is still considered to be a young research field. It is much different from machine translation, where the aim is to understand the purpose and goal of communication and generate a natural language text, which is not template based.

The initial research in the area of NLG has established that NLG is not just reverse of language understanding. It entails issues like lexical choices, text planning, syntactic realization, structure realization etc. The area gained

* Dept. of Computer Science Banasthali University Rajasthan sandhya.singh@gmail.com

** C-DAC, Pune Maharashtra darbari@cdac.in

*** C-DAC, Pune Maharashtra krishnanjanb@cdac.in

**** Dept. of Electronics Banasthali University Rajasthan seemaverma3@yahoo.com

momentum in the decade of 90's. Different architectures, tool and grammars have been designed and implemented since then. Still the aim to design a general purpose NLG system simulating the human like processing of language is yet to reach.

This paper provides a study of different architectures and NLG systems, which have been designed and experimented by the research community. This study is an attempt to create an initial baseline for researchers in the field of NLG to understand the area of NLG and NLG systems, the approaches which have already been experimented and the kind of results obtained from them. The study can assist in deciding the future path in the area of NLG systems. The paper is organized as follows: section 2 gives a brief about NLG system and its architecture, section 3 provides a brief about the available downloadable NLG systems designed and experimented by the research community, section 4gives a firsthand experience while using these NLG systems, section 5 provides an analysisof these tools in the terms of completeness, domain, language used to design, grammar used, usage experience, input and output of the system, closeness to true NLG system etc., section 6 introduces the idea of true NLG system and section 7 concludes the paper with futuristic view.

## 2. NLG SYSTEM AND ITS ARCHITECTURES

Natural language generation is a sub field of natural language processing which deals with task of generating natural language output through a computer representation of knowledge base. Researchers view it as the inverse of NLU where the computer is deriving the meaning from the input sentence. A NLG system is a computer software system, which can automatically generate a human understandable text using language knowledge and context of communication keeping the goal of communication in knowledge. To generate the required output, different algorithms and representations are used for tasks like document planning, content selection, Microplanning, linguistic realization, structure realization etc.

NLG researchers have to deal with generation from a variety of non-linguistic representations like sensor device data, database entries, scientific formulae and other expressions of knowledge representation. The generated text has ranged from one-word responses, single sentence to multi sentence text. Researchers are also exploring the idea of multimodal output which combines natural language text with non-verbal expressions like gestures, graphics etc.[1].

The architecture of NLG system describes the structure and behavior of the system to generate natural language output with a specific goal using the knowledge representation. Modularization encourages reuse of modules across applications and makes it simpler to work on specific part as the need be. Broadly, following architectures have been experimented by NLG researchers community [2]

1. A linear pipeline architecture with the modules like document planning, Microplanning and surface realization as given in Fig -1*a*. This architecture is most experimented by the NLG community.
2. Another pipeline architecture with different modularization like document plan, phrase specification and text specification as given in Fig-1*b* is also suggested by the community.
3. Feedback architecture where feedback between stages is possible to improve upon the final outcome as given in Fig-1*c*. This architecture focuses more on tuning each stage as per the goal of generation.
4. No modularization architecture where all decision making task and constraint satisfaction task is done in parallel as given in Fig–1*d*. This architecture proposes all tasks in parallel with no assumption about which task is processed first.

The generation architecture is required to transfer the knowledge from knowledge systems to generation specific task which has to be more adaptive to heterogenous knowledge search. The challenge lies in identifying the standard knowledge structure for processing.

## 3. OPEN NLG SYSTEMS

AnOpen NLG system is a computer program designed by NLG researchers for experimentation and is openly available for the community to download it, use it, to understand its functioning better.


**Fig. 1*a* – Pipeline Architecture ,**

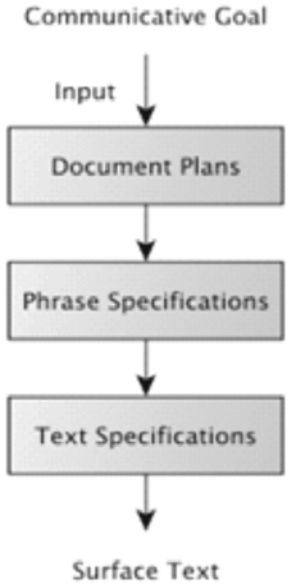


**Fig 1*b* – Pipeline Architecture with different modules.**

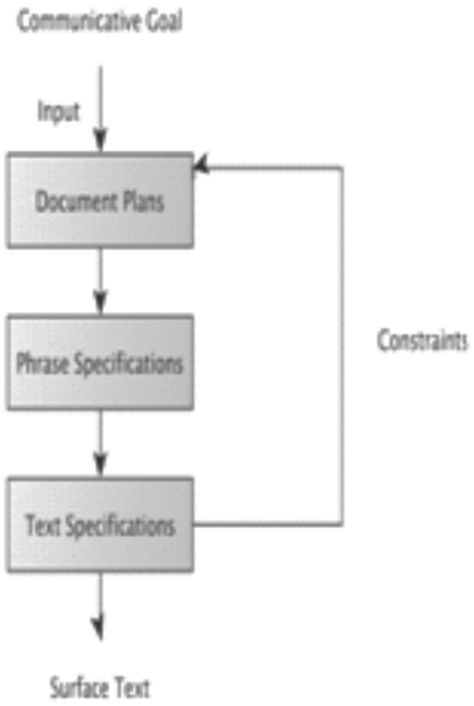


**Fig. 1*c*-Feedback Architecture.**

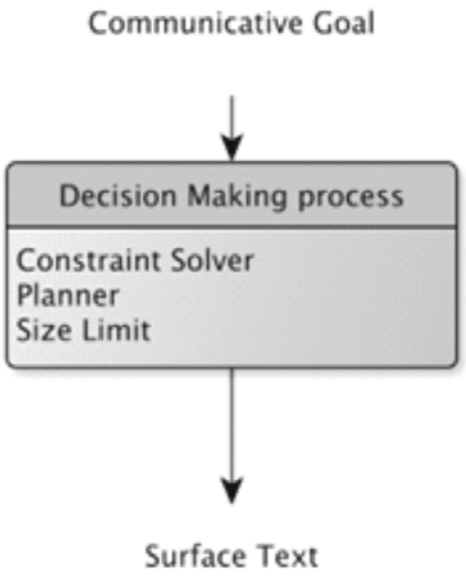


**Fig 1*d* – Parallel Architecture.**

The idea behind these kind of open systems is to serve and share with the community the progress, which has been already tested. The young researchers in the area can benefit from these results and focus specifically on the areas, which needs more attention.

The true NLG system is expected to generate language depending on the understanding of input data, context of communication, the kind of user and the goal to generate text. Besides these the NLG system is also expected to know the grammar and structure of the output natural language. The structure of natural language is an ambiguous and redundant feature with multiple possible variations. These features of language understanding can be rooted to grammar rules, studies related on how people write and analysis of corpora.

For the computational aspect of NLG, there can be a module related to content selection, sentence structure, lexical choice, morphology selection with a systematic discourse structure depending on the architecture and approach followed. The idea behind modular construction of an NLG system is to be able to handle each subtask, regarding parameters, constraints and selections related to text generation with ease from software development aspect. The systems considered here for this survey is listed as downloadable NLG systems[2]on ACLWeb. It lists out all open source NLG systems and grammars which are openly available. For this paper only complete NLG systems have been considered. The details of these systems are referred from the research publications based on these experiments and the project sites of these systems. Followed here is the study of complete (system implementing all 3 modules as per the architecture suggested by Reiter & Dale[2]) NLG systems in terms of purpose, language used to design, architecture followed etc.:

### A. ASTROGEN (Aggregated deep and Surface naTuRal language GENerator)

A prolog based system designed byHercules Dalianisbetween 1996-1999. It was originally implemented for telecom domain but can be used for multi-domain application with support for basic domain specific lexicon. ASTROGEN system contains two modules – Deep generator and Surface generator.The Deep generator module consists ofa multiplesub-modules for doing aggregation.The Surface generator module is based on DCG surface grammar (Definite Clause Grammar) where the lexical items are generated. [3].The ASTROGEN system can be used with other systems by converting its representation into f-structure (functional information structure). ASTROGEN only contains a sentence planner, which performs aggregation based on the rules, which assists in removing the redundant portion of text without changing the content. It does not contain a separate text planner[3].

As per the architecture shown in fig. 2, the system takes as input a set of f-structures (an internal representation) based on the content selected and performs a sentence planning.On this structure the aggregation rules are applied followed by pronominalization.Then a coherent discourse of the f-structuresis created out of this aggregated data[3].

With this as input, the surface generator module produces the syntactic surface structure and the lexical objects. Finally, the post-processing of the text is done using the sentence transformer [3].

### B. CLINT

CLINT is a Hybrid Template/Word-based Text Generator designed by R. Gedalia and M. Elhadad between 1999–2000 using C++ on windows. CLINT system combines two techniques for text generation[4]:

- Templates
- Word-based generation for Noun phrases

It consists of four main components[4]:

- A template development system
- A problem definition system
- A Noun-phrase generator
- The run-time generator

The purpose of problem definition module is to link templates with a decision tree. While running the system, the run-time generator module presents some questions to the user based on the decision tree created in the problem definition module and assigns the values to the parameters.

The decision tree is traversed and each leaf is assigned a template,which is initialized with the value of the parameters.

These parameter values are given to NP generator which decides what NP form is more suitable based on the discourse context at each point. The runtime generator takes input from the user and database to collect information on each specific case and accordingly selects the most fitting template and completes it with the data collected. It then calls NP generator with this completed template and produces the complete text as per the discourse context[4].

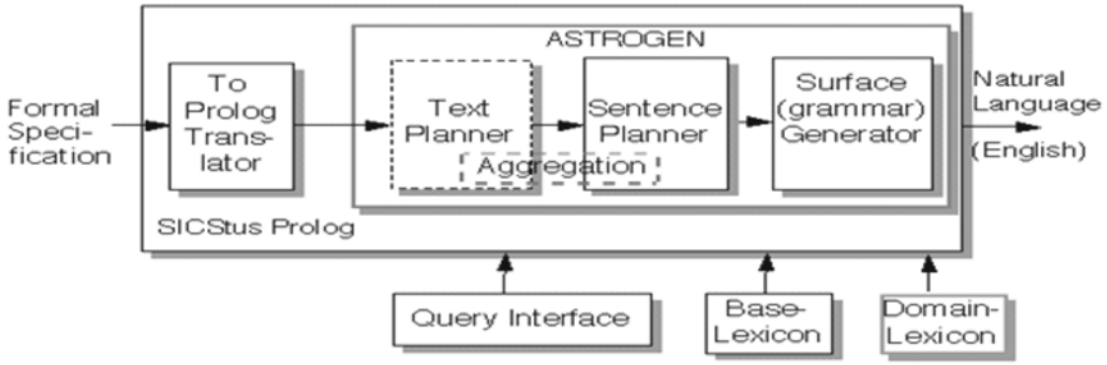


**Fig. 2. ASTROGEN architecture [3].**

## C. DAYDREAMER

A system designed by E.T. Muller between 1983-1988 using Lisp.DAYDREAMER is a recursive-descent generator for reporting stream of thought. The author has defined daydreaming as a process of fantasizing personal ambitions and thoughts of the past or future. It is assumed that daydreaming [4]:

- Enhances performance by utilizing the free time by acquiring from past experiences and utilizing these experiences into future tasks.
- Enhances innovativeness by visualizing imaginative scenarios and identifying analogies between the tasks.
- Assists in controlling emotions.

Based on the input, the system produces English descriptions of its emotional states, associations, recalled past events, imagined variations of past events, and imagined future events. Daydreamer works in the domain of interpersonal relations and common issues related to job, dating and entertainment[4].

## D. Multimodal Unification Grammar

Multimodal Unification Grammar Workbench is a project by David Reitter, which started in 2002 using Prolog. The project is an ongoing project. MUG Workbench is a framework for developing a Multimodal NLG system which works onFunctional Unification Grammar (MUG). It is designed to do three things, which are done in parallel [5]:

- Multimodal Fission which distributes output to different interaction/communication modes.
- Sentence planning which decides what information to include in the final utterance.
- Natural Language and graphical user interface which produces the output.

Technically, the MUG system consists of several components like [5]

- Grammar formalism derived from the combination of functional, weakly typed attribute-value-matrices.
- A hybrid generation algorithm using soft and hard constraints, which can model the efficiency the output.
- A central realizer, which can optimize different constraints using iterative-deepening branch & bound, depth-first search algorithm.
- A knowledge base.
- MUG Workbench including several visualization options like LaTeX.
- The FASiL VPA grammar.

## E. NaturalOWL

A project started byDimitrios Galanis and Ion Androutsopoulos since 2007 onwards using Java. Natural OWL is a multilingual natural language generation system, which describes instances and classes in the linguistic form based on the ontological relationsexpressed in RDF[6]. The pipeline architecture of the system is given in fig. 3.

In document planning stage, the system chooses OWL triplets from the ontology, which is to be conveyed to the user. In micro-planning stage, the systemcreates sentences using each selected fact through micro-plans.In surface realization stage, the final text is produced with tags marked automatically to indicate grammatical categories, punctuation symbols etc. [6]. The domain-dependent generation resources help NaturalOWL produce significantly better texts, and creating these resources using ontology that the resources can be constructed with relatively light effort [7].

### F. NLGen

A project by Ben Goertzel started in 2009 using Java. The NLGen natural language generation system uses the SegSim algorithm for generating English sentences[6].The SegSim algorithm divides a collection of sentences into words and matches it in memory to find similar words with known linguistic expression [7].

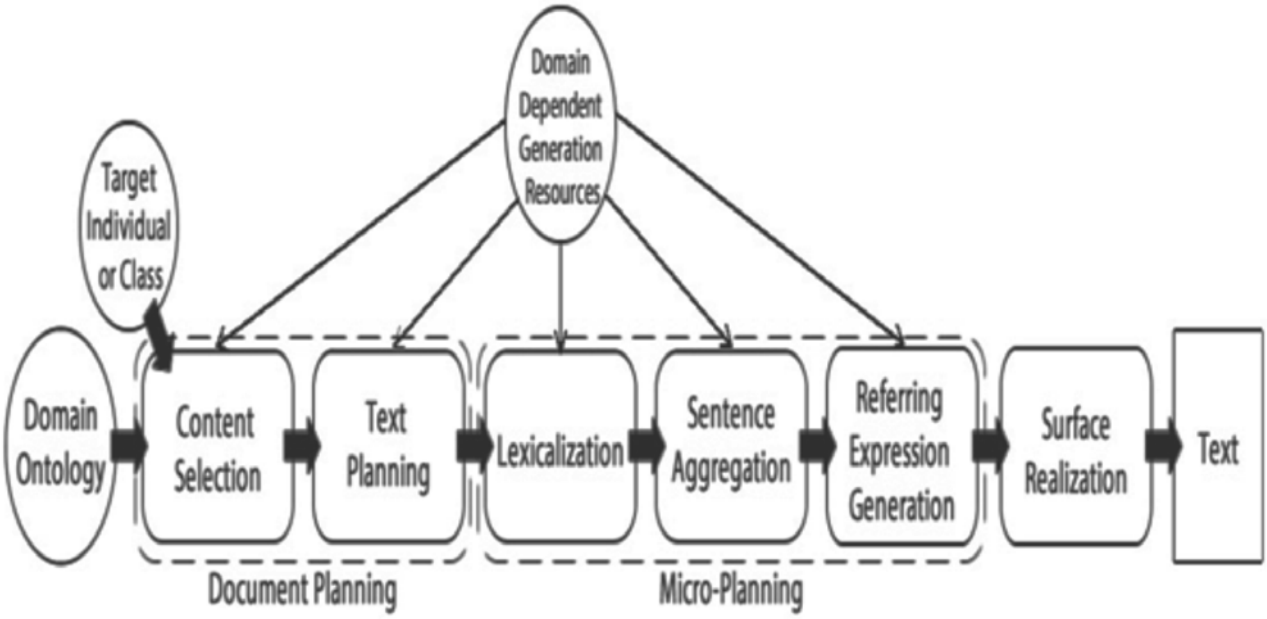


**Fig. 3. Architecture of NaturalOWL system [6].**

The sentence construction is based on the statistical analysis of output of RelEx[8], which is an English-language semantic dependency relationship extractor. It uses a collection of grammar rewriting rules to tag the subject, object, verb and many other dependencies between the words of a sentence.Link grammar parser is used for extracting the linguistic relationship.NLGEN does the inverse, generating natural language text from semantic relations in the same format. It is a part of OpenCog project [8].

### G. NLGen2

A project started by Blake Lemoine in 2009 using Java.The NLGen2 natural language generation system is based on RelEx[8] dependency parser, along with Link-Grammar linkages to generate English sentences as output[9].

It is also a part of OpenCog Project. The NLGInputView module converts the input sentence into a propositional form. This is then processed foroutput sentence based on Levelt's theory of generation, the Minimalist Program, and Simpler Syntax respectively. NLGen2 based on these theories is an attempt to achieve linguistic possibility. The architecture of NLGen2 is given as [9]:

NLGen2 architecture acts as a foundation for a research programthat is based on a robust incremental language generation module.

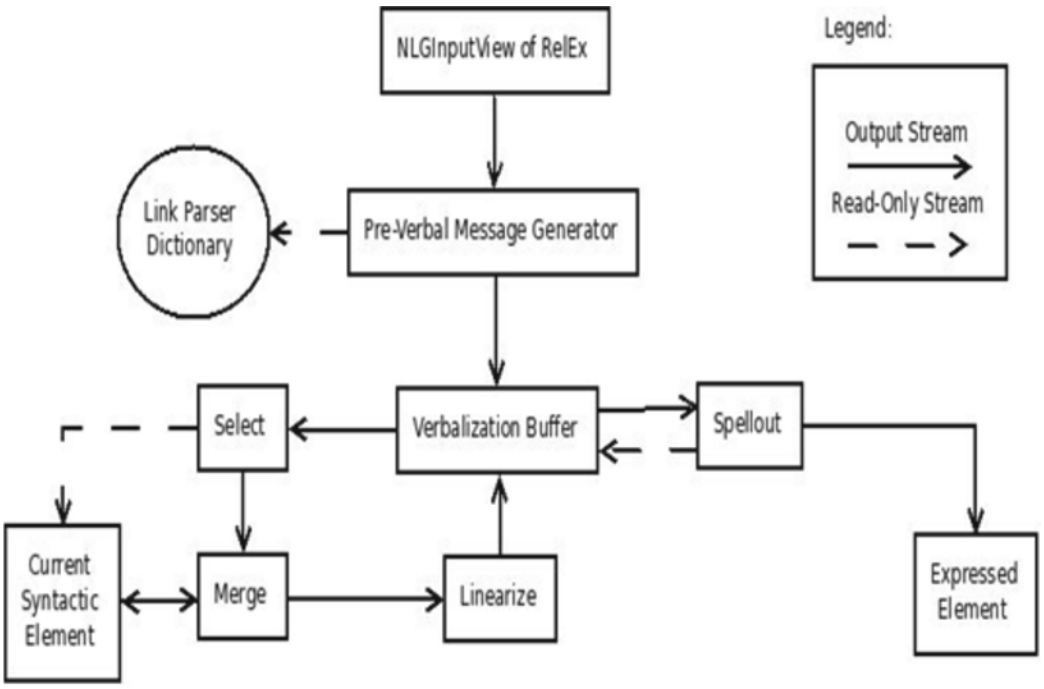


**Fig. 4. General Architecture of NLGen2.**

## H. STANDUP(System To Augment Non-speakers Dialogue Using Puns)

A system designed by Waller A. et al. during2003-2007 using Java.The STANDUP project was designed for generating simple jokes using an interface meant for non-speaking children[10]. Psychologists have explored and proved that humor can be an effective way to help non-speaking children to learn to use language more effectively [10] [11].This system is implemented as a program with an interactive user interface, designed for children with complex communication needs (CCN) which allows them to experiment with the construction of simple jokes. The STANDUP project is a kind of language exploration system meant to assist in Linguistic skills, vocabulary development and social communication skills of children. It consists of three stages of processing as schemas, description rules, and templates for joke generation [10].

## I. Suregen-2

An ontology based project for medical domain by Dirk Hüske-Kraus started in 2002 using Lisp.SURGEN-2 is a hybrid, ontology based system, which can be used for the generation of medical documents consisting of findings, surgical reports and referral letters. It follows a hybrid approach, combining the stored text phrases (with variables) and bottom up generation. The sub-modules included are[11]

- A surface generator,
- A predefined open ontology for medicine,
- A module for conceptual and sentence aggregation,
- A module to refer to common findings based on reports,
- A feature to refer to the common medical terminology and phrases,
- A module meant for the generation related to semantic functions,
- A method to deal with synonyms, antonyms, cohyponyms etc. that is required for aggregation.

It is designed as a command based system, which can be used to build another NLG system or a complete system in itself for clinical report generation [12].

### J. TG/2

A template based system started by Stephan Busemann, in 1998 using LISP. TG/2 is a shallow verbalizer that can be tuned to new domains and tasks. It can combine context-free grammars with templates and stored text in a single output. So the language model used by this system depends on the type of application [13]. It is very similar to shallow parsing.

Generation rules of TG/2 are defined as condition-action pairs. TG/2 is an example of template-based generator. The system separates the generation rules from its interpreter.

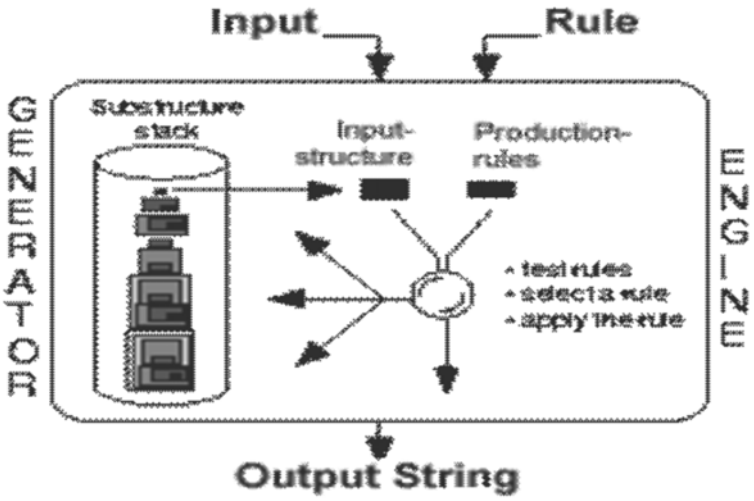


**Fig. 5. Architecture of TG/2 system[14].**

**The interpreter consists of the standard three-step processing cycle [14] :**

- Determine the set of applicable rules,
- Select one from this set, and
- Execute it.

## 4. FIRSTHAND USER EXPERIENCE

After theoretical study of open source complete NLG systems, the next logical step is to use these systems to get a deep insight aboutthe input, output and the support grammars needed. This study can provide us with a fair view about the positives and negatives of these systems. It can help in identifying - how true these NLG systems are? And also give some insight in the direction of research towards a true NLG system.

**To get a firsthand experience of these NLG systems, the approach adopted was :**

- Configure the system with the mentioned Software's.
- Download the NLG system from their respective links.
- Build and prepare the systems on machine for usage following the ReadMe instruction guidelines.
- Test the system with inputs and analyze the output generated.

**The overall experience has been quite varied.The commonhindering experiences while testing these systems are listed as :**

- Version issues of the configuring software and the one mentioned in the guidelines of NLG system software.
- Version issues of the supporting grammar libraries.
- Specifically with the completed project, the configuration required was not readily available.
- In one of the case, the system could not be tested, as machine was not able to support the software (the required operating System was not supported by the current hardware in use).
- In other case, the link provided was a dead link, but the project is an ongoing project.

**A detailed experience with considered NLG systems are briefly discussed here :**

## A. ASTROGEN

The software is easily available on the ASTROGEN site in a zip file format but no recommendation about the version compatibility of SICStus prolog mentioned on the site. Since the project is Complete, no online support is available. Though the site is comprehensively designed with detailed information about the type of input and output. It also explains the different configuration, which can be experimented with this system.

Being prolog based, the input has to be given in a predicate function format [3] *e.g.*

- Paraphrase (+IN) where IN is an input of frametype, which results in an NL string.
- Paraphrase (+IN, -NL) where IN is an input of frametype, which results in a list of NL.
- Input IN = *f*(T,P, Arg1, Arg2, Arg3) where T = {past, present, future} is the time predicate , P is arelation predicate verb *e.g.* are, have etc. and Arg1, Arg2, and Arg3 are subjects like nouns, pronouns, *e.g.* lexical items like John, subscriber etc.

Different switches are recommended for different kinds of need like lexical aggregation, syntactical aggregation, etc. The input should tell the words in the output sentence and the kind of aggregation to be applied along with POS information, and then it can generate a fluent sentence. But the user should know what kind of output is needed, accordingly select the rules and word.

The system is able to generate the sentences provided the lexical choices, rules to be applied are provided. The experience is still far from a true NLG system.

## B. CLINT

This project was designed on windows 3.1 platform, so it cannot be tested on present day machine due to compatibility issues. Also this being a completed project not much of technical support is present. From the website literature, it can be inferred that being template based system for business purpose the input has to be provided in the standard fill in the template format with limited variation. It could generatesimple standard strings needed for business domain, but could not match to the expectations of a true NLG system.

## C. DAYDREAMER

This is the oldest system from those listed here, which got completed in 1988, not much could be discussed about its hands on experience. But being based on common Lisp, and the discussions available in the research papers, it also could not live up to the expectations of True NLG system.

## D. MUG

This is a Multimodal system compatible with SWI prolog 5.4.0 or later versions. The instructions given in the supporting ReadMe file is sufficient to get started with the system. It is an ongoing project, so the site is well supported for any issues and bugs.

Being Multimodal it can accept input through different mechanism and provides the ease of transfer among output generation. It implements one grammar rule across modalities so the output is coherent. It generates multiple outputs with scores for each. The user can select the formalism based on the scoring function known as fitness function.

## E. NaturalOWL

This NLG system has been designed as plugin for ontology development tool called protégé. Though the ReadMe instructions are available, the protégé version has changed since it was launched first; there are certain discrepancies in the documentation and actual system installation. Online support for protégé is available but NaturalOWL needs some struggle before a smooth execution. Java version and protégé version compatibility has to be kept in mind while using it.

Being an ontology based system, generation is possible if the domain ontology exists or created in proper order. This system implements generic architecture with all the sub-modules described. To generate a sentence, it selects the logical facts from the ontology graph and are ordered based on the ordering annotations. The Microplanning stage is implemented with template kind of approach using RDF notations. It is then converted in a sentence based on the lexicons from the domain ontology.

It requires users to select the facts to be included in the sentence, and also the lexicon selection is based on the ontology. The output of this system is comparatively better than other system but lot more scope towards true NLG system.

### F. NLGen

This being a part of ongoing OpenCog project, it is well supported on technical aspect. The challenges occur due to discrepancies in the version issues of java and the link grammar used by this. It translates RelEx relationship-sets into link parses leading it into sentences. The current versionworks only for simple sentences. The input has to be provided in relationship sets between words along with POS.

### G. NLGen2

NLGen2 is also a part of OpenCog project, which is still under process. This also provides the basic version issues between java, link grammar parser which is the knowledge base used. It accepts the input in propositional format and converts it into a preverbal message followed by series of merging, linearization to generate a simple sentence.

### H. STANDUP

This being a template based system for generating text for non-speakers. It provides an interactive interface for selection, which can lead to generation. It can generate simple sentence that utilize the ambiguous meaning of words from the language resource like wordnet and uses it for pun generation. It was easy to test due to web interface provided with the system.

### I. Surgeon-2

A system designed in Lisp, based on ontology for medical domain. Though based on classical architecture, it does not implement all the sub-modules listed. Being an ongoing project proper support is available for the users. This system is a acceptable attempt to generate simple sentence from medical domain. But the aim of true NLG system is still far.

### J. TG/2

This system is temporarily unavailable for usage and testing so nothing can be commented on its usage and ease of working and its closeness to true NLG system.

## 5. ANALYSIS OF DISCUSSED NLG SYSTEMS

**After the theoretical discussion of open NLG systems listed [1] and the first hand user experience, these are further analyzed on parameters like :**

- Is it supporting domain independent generation?
- Is the language chosen to design has any role to play?
- Is the project complete or ongoing?
- Architecture implemented by the system.
- What is the behavior of the systems if same lexical input is given?
- What kind of output knowledge representation followed?

Table-I below shows a tabular description of all the NLG systems studied here with a focus on their timeline, the language used, their domain and architecture used etc.

Out of 10 NLG systems discussed here, ASTROGEN, Natural OWL, NLGen, NLGen2 are using the classical architecture, CLINT, STANDUP, TG/2 are template based, DAYDREAMER follows a feedback architecture, MUG follows the multimodal architecture and Surgeon-2 uses hybrid architectureclassical architecture, CLINT, STANDUP, TG/2 are template based, DAYDREAMER follows a feedback architecture, MUG follows the multimodal architecture and Surgeon-2 uses hybrid architecture.

On the programming language technology front, 5 are using Java which has a rich library base language processing libraries, 4 are implemented using traditional AI programming languages, and 1 is implemented using a very strong language as C++.

**Table 1 : A tabular analysis of Open NLG systems.**

| System Name | Architecture Used | Language Used | Ongoing/ Complete | Approach Followed | Domain | Duration |
|---|---|---|---|---|---|---|
| ASTROGEN | Pipeline | Prolog | Complete | Aggregation approach | Independent | 1996-1999 |
| CLINT | Template Based | C++ | Complete | Customized approach | Business Letter | 1999-2000 |
| DAYDREAMER | Feedback Based | LISP | Complete | Recursive approach | Interpersonal | 1983-1988 |
| MUG | Multimodal Architecture | Prolog | Ongoing | Multimodal functional unification grammar | Personal Assistant | 2002 onwards |
| NaturalOWL | Pipeline | Java | Ongoing | Ontology based | Domain Ontology | 2007 onwards |
| NLGen | Pipeline | Java | Ongoing | Segmentation & similarity approach | Independent | 2009 onwards |
| NLGen2 | Pipeline | Java | Ongoing | Levelt's theory of generation | Independent | 2009 onwards |
| STANDUP | Template Based | Java | Complete | Standard Wordnet based templates | Pun Based | 2003-2007 |
| Suregen-2 | Hybrid | LISP | Ongoing | Ontology based hybrid approach | Medicine | 2002 onwards |
| TG/2 | Template Based | Java | Ongoing | Template based | Independent | 1998 onwards |

In terms of knowledge base used, NLGen, NLGen2 are OpenCog project which uses link grammar parser and other resources under OpenCog project. NaturalOWL and Surgeon-2 uses ontology as knowledge base. Besides these, other knowledge bases, which are being designed and used, are Functional Unification Formalism (FUF), KPML, LKB, CCG, wordnet etc. The knowledge bases listed here are not comprehensive, but yet it needs to be explored further.

While experimenting with same lexical input to these systems, the overall experience was very versatile. The input format varied for all systems even though the lexical choice was kept similar for all. ASTROGEN system needed the list of all words to be kept in the output sentence with tense specified explicitly in a predicate function format.For NaturalOWL, example ontology was used to generate the sentence by selecting the necessary facts needed in the output. NLGen and NLGen2 system required the input for which relationship set existed which limited the scope but resulted in simple output sentences. STANDUP generated the sentences based on the ambiguous words selected from wordnet using a fix template. Other systems could not generate any output due totechnical constraints.

From first hand user experience, it is evident that these systems have managed to generate simple sentences with the help of grammar and ontologies but these are still far from general purpose usage. A lot of focus is needed on simplifying the user input. The grammar should be vast enough to generate complex sentences.

## 6. VISION TOWARDS TRUE NLG SYSTEM

This study was done to understand the idea of true NLG system. A true NLG system is expected to generate text with human kind of ease with word usage and fluency in the areas like data- to- text report generation, coherent text summarization, image description, robotic communication, language teaching, debating etc. The requirement of these application areas deals with a vast understanding of language grammar along with the domain information, its ability to interpret the information and produce it in a fluent language form. The expectations from the true NLG system is to understand the communication goal and the target audience, use the grammar of the language along with the domain knowledge and the context of communication thus generating the text response for the user which is concise and coherent in approach.

## 7. CONCLUSION AND FUTURE WORK

This study provides an overview of NLG, its architectures and a description of different open NLG system. It presented a comprehensive exposure to open NLG system with a focus on its architecture, approach used for generation and language technology used to create these systems. The first hand user experience has given enough insight on the probable direction of approach. This study has benefited us by providing a basic insight into the NLG systems designed and experimented so far and their strengths and weakness, which can be used as a guideline towards experimenting with a new approach.

In future, the plan is to use these insights and the vision of true NLG system todesign a novel approach, which can lead us towards the aim of designing a true NLG system, which can be used for text summarization, image description and many other areas of language generation.

## 8. REFERENCES

1. Stock, Oliviero, Elena Not, and Massimo Zancanaro. “Intelligent interactive information presentation for cultural tourism.” Multimodal intelligent information presentation. Springer Netherlands, 2005. 95-111.
2. Reiter, Ehud, Robert Dale, and Zhiwei Feng. ”Building natural language generation systems.” Vol. 33. Cambridge: Cambridge university press, 2000.
3. Dalianis, Hercules. “Aggregation in natural language generation.”Computational Intelligence 15.4 (1999): 384-414.
4. Mueller, Erik T. ”Daydreaming in humans and machines: a computer model of the stream of thought.” Intellect Books, 1990.
5. Reitter, David. “A development environment for multimodal functional unification generation grammars.” ITRI-04-01 INLG04 Posters: Extended(2004): 32.

6. Galanis, Dimitrios, et al. “An open-source natural language generator for OWL ontologies and its use in Protégé and Second Life.” Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics: Demonstrations Session. Association for Computational Linguistics, 2009.
7. Galanis, Dimitrios, and Ion Androutsopoulos. “Generating multilingual descriptions from linguistically annotated OWL ontologies: the NaturalOWL system.” Proceedings of the Eleventh European Workshop on Natural Language Generation. Association for Computational Linguistics, 2007.
8 Hart, David, and Ben Goertzel. “Opencog: A software framework for integrative artificial general intelligence.” Frontiers in Artificial Intelligence and Applications 171 (2008): 468.
9 Lemoine, Blake, and Lafayette UL. “NLGen2: a linguistically plausible, general purpose natural language generation system.” (2010).
10 Manurung, Ruli, et al. “The construction of a pun generator for language skills development.” Applied Artificial Intelligence 22.9 (2008): 841-869.
11 Binsted, Kim, Helen Pain, and Graeme Ritchie. “Children’s evaluation of computer-generated punning riddles.” Pragmatics & Cognition 5.2 (1997): 305-354.
12 Hüske-Kraus, Dirk. “Suregen-2: A shell system for the generation of clinical documents.” Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 2. Association for Computational Linguistics, 2003.
13 Busemann, Stephan. “Best-first surface realization.” arXiv preprint cmp-lg/9605010 (1996).
14 Busemann, Stephan. “Ten years after: An update on TG/2 (and friends).”Proceedings of European Natural Language Generation Worksho.