International Journal of Engineering Research and General Science Volume 3, Issue 2, Part 2, March-April, 2015
ISSN 2091-2730

Natural Language Generation

Girija Godbole

Student of Computer Engg at PICT college of Pune University,

gsgodbole@gmail.com , +919823617208

Abstract— Many a times it is needed to produce understandable texts in English or other human languages from some underlying non-linguistic representation of information. This is achieved by Natural Language Generation (NLG). NLG systems have a wide range of applications in the fields of media, medicine, computational humor, etc. In this paper, 'pipeline architecture' is explained which contains steps involved in the process of NLG. Throughout, the emphasis is on established techniques that can be used to build simple but practical working NLG systems.

Keywords— Computational Linguistics, natural language processing, pipeline architecture, natural language processing,

INTRODUCTION

NLG or Natural Language Generation is the process of constructing natural language outputs from non-linguistic inputs. NLG system is a translator that converts a computer based representation into a natural language representation. Natural Language Generation is the inverse of natural language understanding (NLU).NLG maps from meaning to text, while NLU maps from text to meaning. The input to NLG system varies widely from one application to another.

The existing NLG systems include FOG- generates textual summaries of weather forecast since 1992, STOP system- produces a personalized smoking-cessation leaflet, ANA generates summaries of stock market activity, LFS, SUMGEN, TEMSIS, TREND, etc. In this paper, we describe the process of NLG, the basic steps involved and the algorithms used to carry out these steps.

The first architecture of NLG was proposed in 1980's which consisted of only two stages- text planning and linguistic realization. Later in 1990's an advanced model consisting of three major steps- Document planner, Micro-planner and Surface Realizer was proposed which proved to be useful in building proper working NLG systems. In this paper, a modified version of this architecture is explained which consists of six steps- content determination, document structuring, lexicalization, aggregation, referring expressions generation and linguistic realization.

The structure of the paper is as follows- the design and analysis of the system which gives an idea about the architecture of NLG in detail, the discussion on implementation results, the conclusion and future enhancement of the topic and the references.

DESIGN AND ANALYSIS OF SYSTEM

Inputs and Outputs of NLG:

Language as goal-driven communication:

NLG is often viewed as a goal-driven communication. The generation of any form of text (word or sentence) is seen as an attempt to satisfy some communicative goal. Communicative goals include informing the hearer of something, requesting or persuading the hearer to do something and of obtaining some information from the user.

Inputs of NLG:

In general, the input set of any NLG system can be thought of a four-tuple (k , c ,u ,d) where, ‘k’ is 'knowledge source', ‘c’ is 'communicative goal', ‘u’ is 'user model' and d is 'discourse history'.

1) knowledge source(k):

It may be represented in different ways in different systems depending upon the nature of the host application eg, one system may use tables of numbers or other may use information encoded in some knowledge representation language. So, there is no specific defined representation for k. But, in a broad sense, we can categorize it in two cases: 1.) information is precisely defined already(directly use k) 2.) information is to be selected from a vast k (part of k is to be used).

2) communicative goal(c):

c depends upon what type of k we are using. For eg if it is of type 1 as mentioned above, then, as the information to be output is already defined, 'c' will be just 'express the specified information'. If k is of type 2, 'c' will be ' express the relevant information'.

3)user model(u):

It is the identification of the intended audience for whom the text is to be generated. The consideration of u is important because the output may depend in the technical knowledge of the user. For instance, if the user is novice, then the output should also contain explanations of technical terms.

4) discourse history(d):

It is keeping track of whatever information the system has given to the user so far. This is generally useful in dialogue systems and single-interaction systems. For other applications, it is kept null.

Output of NLG:

The output of any NLG system is fragments of text. Fragments are the collection of sentences. The length of text may differ from application to application. For example, in dialogue systems, output is only one word 'yes' or 'no' whereas in single-interaction systems, the output may be of several pages.

Architecture of NLG:

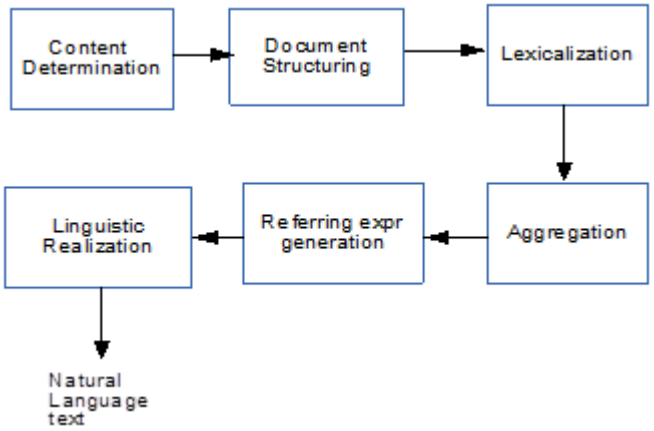

fig. 1. Pipeline architecture of NLG

STAGES OF THE PIPELINE :

Content Determination:

It is deciding what information is to be communicated in the text. Either the host application will tell what information is to be taken or the NLG has to itself figure it out and select the relevant information. The choice of what information is to be expressed depends on the following factors:

1) The communicative goals may require different information to be expressed eg if NLG describes weather over a period and also gives explanation of different technical terms, then different content will be required in each case.

2) It depends on the user or hearer. If he/she is novice, then more explanation is needed.

3) There may be constraints on output eg, the text is needed to be fit only within the given area.

4) It depends on the underlying information source. What is worth saying will depend on the nature and content of information available.

Ultimately, what information is to be included and circumstances under which it is to be included is completely dependent on the application.

Document Structuring:

Structuring is important because an unstructured text can be thought of a story which is starting in the middle, then has the conclusion and then the introduction; which is meaningless. Structuring is nothing but proper sequencing of sentences in the text. There are three approaches:

1) Schemas : They are templates which explicitly specify sentence ordering and grouping for a document. Schemas work well in practice for texts which are short and/or have a standardized structure, but have problems in generating texts which are longer and do not have a fixed structure.

2) Corpus-based: use statistical corpus analysis techniques to automatically build ordering/grouping models.

3) Heuristic-based: structuring is based on heuristic rules which can come from theories of rhetoric, psycholinguistic models, etc.

Lexicalization:

It means choosing the content words like nouns, verbs, adjectives and adverbs for the generated text. There are two types:

1) Conceptual Lexicalization: This includes slight modification of information. The words are slightly changed or at sometimes even the completely new word is used.

2) Expressive Lexicalization: It gives the output as it is with correct verbs, nouns, etc.

For example, we have three words- 'male', 'unmarried' and 'adult'. Now, using expressive lexicalization, the output will be- 'unmarried adult male' . But, with conceptual lexicalization, the output will be- 'bachelor'.

Aggregation:

It is a process in which two or more sentences are merged to form a single sentence. Aggregation helps to build sentences which communicate several pieces of information at once.

Algorithm of aggregation:

k-way hyper-graph partitioning:

Input is a conceptual graph. A conceptual graph is a set of propositions or sentences. The vertices indicate propositions. Each hyper edge connects one or more proposition. The weight of each hyper edge is found by Context Sensitive Discrimination Model. After this, the hyper graph with edge weights is the input to multilevel k-partitioning algorithm.

Referring Expressions Generation:

Any domain consists of entities. Entities are things that can be talked about whether concrete or abstract. Referring expressions generation is concerned with how we produce a description of the entity for the hearer in the given context.

When an entity is mentioned for the first time, it is called 'initial reference' and after that, it is called 'subsequent reference'. The content of the referring expression is determined with the help of discourse history. Once we have determined which properties of entity should be used in a description, we still have to consider how those properties should be realized in the output.

Linguistic Realization:

Aim of linguistic realization is to produce a text which is syntactically, morphologically and orthogonally correct. For instance, consider a sentence "It rained for 8 days from 11th to 18th ."

1) Syntactic: The syntactic component of realizer will add the words like 'for', 'from' ,'to', etc.

2) Morphological: The morphological component will produce the past tense of 'rain' and give output 'rained'.

3) Orthogonal: The orthogonal component adds full stop, capital letters at start of sentence, etc.

DISCUSSION ON IMPLEMENTATION RESULTS

For generating simple sentences and paragraphs Simplenlg, a JAVA API was used. Simplenlg is intended to function as a 'realization engine' for Natural Language Generation architectures, and has been used successfully in a number of projects, both academic and commercial. It is a tool to perform two of the above tasks of the pipeline architecture namely 'aggregation' and 'linguistic realization'. The version of simplenlg used is v4.4 which is integrated with Eclipse editor.

The input and outputs are specified in the following table:-

Sr no	function	Input	Output
1	createSentence ()	A sentence eg ("my dog is happy")	Stored in NLGElement object s1
2	realiseSentence(s1)	s1	Orthographically correct sentence ("My dog is happy.")
3	CreateClause() setSubject(subject) setVerb(verb) setObject(object)	Subject, verb, object	A full sentence

Table 1

CONCLUSION

We have thus, described a six-stage architecture for natural language generation. The NLG system architectures have evolved since 1980 till date with significant modifications in the general steps of the method to generate text. Currently, we have used Simplenlg realizer to generate simple sentences and paragraphs which is a JAVA API with inbuilt functions for almost all tasks. The future scope for this implementation will be writing codes ourselves for these inbuilt functions which means not using Simplenlg but writing java classes for carrying out all the NLG functions.

REFERENCES:

[1] A.Ramos-Soto, A. Bugareın, S. Barro, and J.Taboada, "Linguistic Descriptions for Automatic Generationof Textual Short-Term Weather Forecasts on Real Prediction Data", IEEE trans. on fuzzy syst., vol. 23, no. 1, pp 44-57, February 2015

[2] H. Leopold, J. Mendling and A.Polyvyanyy, "Supporting Process Model Validation through Natural Language Generation",IEEE trans. on soft. Engg., vol 40,no 8, pp 818-840,August 2014

[3] H. Banaee, M. Uddin Ahmed, A. Loutfi, 2013,"A framework for automatic text generation of trends in physiological time series data" IEEE Inter. Conf.on Syst., Man, and Cybernetics 1-6

[4] E. Reiter and R. Dale, "Building Natural Language Generation Systems" ,Cambridge, U.K.: Cambridge Univ. Press, 2000.

[5] Paul Semaan, "Natural Language Generation: An Overview", Journal of Computer Science & Research (JCSCR) - ISSN 2227-328X ,Vol. 1, No. 3, Pages. 50-57, June 2012

[6] Goldberg E, Driedger N, Kittredge R, "Using Natural-Language Processing to Produce Weather Forecasts", IEEE Expert, vol. 9, no.2, pp. 45–53, 1994.

[7] Hemanth Sagar Bayyarapu, "Efficient Algorithm for Context Sensitive Aggregation in Natural Language Generation", Proceedings of Recent Advances in Natural Language Processing, pages 84–89, Hissar, Bulgaria, 12-14 September 2011.

[8] Regina Barzilay, Mirella Lapata, "Aggregation via Set Partitioning for Natural Language Generation", Proceedings of the Human Language Technology Conference of the North American Chapter of the ACL, pages 359–366, New York, June 2006. c 2006 Association for Computational Linguistics

[9] Anne H. Schneider Alasdair Mort Chris Mellish Ehud Reiter Phil Wilson, "MIME - NLG in Pre-Hospital Care", Proceedings of the 14th European Workshop on Natural Language Generation, pages 152–156,Sofia, Bulgaria, August 8-9 2013. c 2013 Association for Computational Linguistics

[10] Rafael L. de Oliveira, Eder M. de Novais, Roberto P. A. de Araujo, Ivandré Paraboni, "A Classification-driven Approach to Document Planning", International Conference RANLP 2009 - Borovets, Bulgaria, pages 324–329

[11] Robert Dale, Sabine Geldof and Jean-Philippe Prost," Using Natural Language Generation in Automatic Route Description", Journal of Research and Practice in Information Technology, Vol. 37, No. 1, February 2005 ,pages 89-105

[12] Donna Byron , Alexander Koller ,Kristina Striegnitz, Justine Cassell, Robert Dale, Johanna Moore, Jon Oberlander, "Report on the First NLG Challenge on Generating Instructions in Virtual Environments (GIVE)", Proceedings of the 12th European Workshop on Natural Language Generation, pages 165–173, Athens, Greece, 30 – 31 March 2009. c 2009 Association for Computational Linguistics