Pre-Syntactic Analysis of NL Sentences for Auto Generation of Programming Interface

Gopinath Ganapathy[1]

ABSTRACT

Auto Generation of Programming Interface is a framework that leverages the NLP and Auto Programming concepts and suggests a probable generation of a syntactically valid piece of code transformed from NL description.

This paper addresses the initial phases of the framework that include lexical and morphological analysis. A size conscious lexicon has been constructed for the framework that is compiled from the data collection survey. A powerful yet simple strategy called projection mathematics is employed to map a lexical component to all possible derivations from its stem.

KEYWORDS : NLP, Auto Programming, Lexicon, Morphological Analysis, Projection Mathematics

1. INTRODUCTION

The self generation of Program from a given information by analyzing it through the computational approach or by means of straight forward application of techniques is one of the methodologies of Auto programming. Auto Generation of Programming Interface is theoretical framework (AGOPI) that is conceived of to generate a target program in artificial language from the given description of the problem in Natural Language that needs to go through various phases in analyzing NL descriptions. More precisely, a framework is to be built, whereby the user can specify the procedures that should be translated into a target program.

NL Representation →AGOPI→TP in Artificial Language.

This paper deals with the analysis of the words, extracted from the NL description of the problem, collected in data-collection sheets (Appendix) from around 750 persons, envisaging that a model lexicon must be constructed for AGOPI. Each word is processed so that it can fit into the lexicon as an entity which is further subject to the morphological analysis. This will enable the framework to carry out the parsing of the NL description.

This paper talks about the initial phase of the intended framework which involves collection of terminology (words both technical and general) that could possibly be used in the description which should be independent of special machine features. A data collection survey has been conducted with 750 people mostly of computer user group including novices at different places (mostly educational Institutions) in order to fetch the words usable while representing the problem. The collection is carried out in three full years among different Institutions (to get a large no. of candidates) in order to get the matured vocabulary and terminology in the programming domain. The candidates were so chosen that they had fully or partly working experience with computers in general and programs in particular. Thus collected words are manually processed initially so that they can be deposited into a loosely structured word base which is to be fed as lookup table for the module that forms a part of the project.

[1]Professor and Head,
Department of Computer Science,
Bharathidasan University, Trichirapalli,
Tamil Nadu - 620 023. India.

Presently, to give the word base the lexicon status syntactic categories are introduced to a smaller extent along with the entries.

2. Related Survey

From a domain-based text, the analysis towards recognizing NL in a few applications and information retrieval systems based on dictionaries. is positive approach in AP designs to encounter the input tokens. Using special purpose dictionary for automatic text processing would show better performance in any information retrieval systems. For the work [5] of technical abstract analysis, the data were gathered from a CS library for terms from a domain of technical abstracts to form a dictionary for language processing system. Though more recent theories treat lexicons as afterthought, yet the essence of these theories is in lexicon [10] since every element of the semantic representation of the sentence ultimately derives from the lexicon. Several projects are going on aiming the creation of lexical knowledge base [4]. A rich lexical structure makes it possible to express rules capturing linguistic generalization based on the semantic content of lexical items, rather than relying on general-purpose inference over an encyclopedic KB [1]. So, the task of constructing a lexicon for a natural language is formidable, not only because of the absence of a well articulated theory of what it should contain, but also the enormous number of the words to be dealt with. This works assumes that there could be no MT without a system having adequate words that must have been understood for the translation task. While these kinds of applications are built with semantic features, there are other few applications like syntax parser may need syntactic category (e.g. verb) and the features like infinite, transitive, etc. in their lexical base. A NL generator needs the words to map the internal concepts of the word to the external world. The content and the organization of the lexicon, however, directly depend upon the application to which it is to be attached.

Wilks [12] described a working analysis and generation algorithm which handles paragraph length input. It maps utterances in one NL to a form unambiguously to the other. Simmons describes about the derivation and manipulation of representation of verbal meanings for a subset of English sentence [9]. Similarly [6] specifies semantic network which defines the meaning of the lexical entry. In interactive NL application like SHRDLU [13] the lexicon is injected with world knowledge of a model. The SHRDLU was designed to answer queries concerning the domain of toy blocks which, with a help of lexicon, made deductions about the state of the blocks. Carroll and Grover [3] have described a lexicon development environment with an aim to develop a morphological and syntactic analyzer in English grammar. [8] has given the multilingual semantic lexicon based on Conceptual Dependency (CD) and Discrimination networks (Dnet) in an objected oriented platform providing an efficient retrieval mechanism using hash table. Hence any NL application will have its own conventions regarding the content, organization, and structure of its lexicon to which the approach of the system differs as per its application. So the lexicon is a mandatory for the frameworks like AGOPI for its analysis.

3. Lexico_Morphological View

Building customized lexicons will have duplication of efforts until a rich and powerful lexical database is developed with different linguistic theories [2]. For a chosen application, say AP system, a word data base would need to have all information about a word that is supposed to be relevant to that application, though it is unlikely to be constructed. A lexicon may be expected to

provide information about the style of the word, phonemes, synonyms, and even diagrams; however AGOPI does not need such a elaborate and full fledged dictionaries. This work believes that the irrelevant items, not needed by the system though it is associated to a word, may be isolated. This leads to keeping words and their forms organized well and application friendly.

Like any other lexicon, our word base (See Chapter Appendix A.3) is also loosely structured with no uniform patterns or records. The morphological analysis may assume that all dictionary users are assumed to possess knowledge about morphology, both inflectional and derivative [2]. But the computing system application like AGOPI is not expected to do so. How to represent the various forms of the entry in computationally viable format is the question that makes the morphological analysis a significant component in our lexicon. AGOPI uses the lexicon where the internal representation is to be explicitly transferred to the memory in order to be accessible and suitable to computer program. This approach is supported by the program which employs certain decomposition and manipulation on the entries having the stream of word with the head word at the first and the derivation factors or affixes at the rest. However all the irregular and independent words are stored as it is.

The computer word forms are constructed in a way by concatenation of segments. The set of all segments can be broken down into disjoint subsets [14]. So, a word (W) is more than a sequence of characters; it is considered as a group of components (C) like root, prefix, and suffix.

$$W = C1 + C2 + \ldots Cn \ (n > 0)$$

The word model exhibiting this linguistic principle is capable of determining the size of the dictionary. Hence, AGOPI builds the lexicon with size consciousness where the terminologies are stored in a manner that they are based on keywords and their morphological affixes, in order to reduce the high degree of redundancy. The underlying concept of organization of the simple word base used in the system is realized through the following figure.

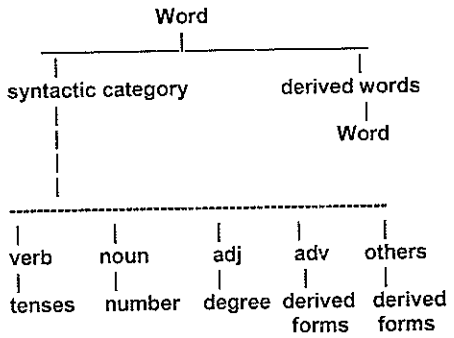

Figure 1: Word Organization

4. Lexical Construction

Since KA is the bottleneck in development of any system, a comprehensive lexicon is very much needed in any automated KA system that involves NL. For example a rule based model for medical expert system deals with lexicon to understand the knowledge contained in English text[7]. As this framework uses a compact and convincing word base to handle English text, grammatical components are easily processed.

The knowledge of words has a dominant role in NLP applications as many lexicons are designed for specific applications like medical DSS [11]. In the envisaged AGOPI model, the words from the survey are collected and manually deposited based on the categories viz verbs, nouns, adverbs, adjectives, pronouns, conjunction, and preposition. The words are to be synthesized based on their morphological features and stored accordingly with the intention that the word has to occur preferably only once in the word base.

In the first category the word types are prepositions, conjunctions and most of the pronouns, which are taken as a full form. The second category is filled up by the

word types like adjectives, adverbs, nouns and verbs which require morphological analysis. (see diagram in Chapter Appendix A-1). Though every entry in the word base of this category follows no uniform pattern, it projects a fixed number of fields depends upon the inflection or the derivation of that entry.

A pattern, for example,

:VERB TENSE_FORMS NOMINAL_DERIVATIONS PLURAL_FORMS ADJECTIVE ADVERBIAL_FORMS PREFIXED_FORMS:

will take values like

:approximate -+ -ion *s ly mis~=:

(See next section for rules of projection.)

Similarly another pattern keeping noun as head word, as in

:NOUN PLURAL_FORM ADJECTIVE ADVERB:

In another pattern, the adjective forms the key stem as in

:ADJECTIVE ADVERBIAL_DERIVATION NOMINAL_FORMS & PLURAL_FORMS PREFIXED_FORMS:

which will assume words like

:common ly ness ality un~=:

For the sake of management, the words that are same for verb and noun (for eg. call, attempt, input etc.) are treated as verb only though they are semantically different. This work is not concentrating on lexical semantics and on complete syntactic analysis, as right now the work restricts itself to pre-syntactic stage and partial syntax analysis only.

It is somewhat clear in our application that the lexicon is not expected to give the synonyms, antonyms and other such features. It is not a pure MRD or a full fledged lexicon for machine translation. Nevertheless it is more than a mere lookup table for pattern matching. It should possess the syntactic features and to some extent the lexical semantics.

This section hints about the implementation phase along with a front end for accessing the word base. Generally, to compromise the speed of the program which exhaustively does the morphological analysis is another weak strategy in MRD usage since the developers are convinced with the reduction of size of the lexicon and the size of the memory, giving less attention to the process optimization. Since the morphological features are introduced in our word base itself, the computation is minimized to a greater extent. At the same time the size of the word base is taken care of so as to occupy less space in the computer storage. The transferring of the word base to the memory based on the projection rules is carefully performed by the program. AGOPI is working with a confined domain of English words and its word base perfectly fits into RAM, Another feature of the algorithm is that when the memory to be filled up by the word base entries, the projection rules are employed only to the needed portion of the word base due to indexing mechanism so that the search and match operations are simplified and the searching time is expected to be reduced to some extent.. The general concept of the logic of the pre-syntax module includes the following core idea.

1) Interfacing screen: It is a GUI for accepting the textual information. (either on-line or off-line).

2) Tokenizing the text: (void tokenizer()) in order to enable the morphological Analysis. (Tokens are stored in a buffer which dies when the program quits).

3) Two-level Lexical match: The token matching is carried out with the following straight forward logic.

a) Direct Token Match (DTM).: i.e. the word from the user's representation is matched directly with the lexical root item.

b) Synthesized Token Match (STM):This is applicable when DTM fails.

i.e. the word read is compared with the expanded lexical array. The lexical root item and its morphologically associated tokens are collectively named lexical array or lexical row. At STM level the lexical array corresponding to the scanned word from the input text should be expanded so that the matching is carried out. Obviously, if both levels fail, the logic reveals the non availability of the item of interest. The logic so designed that the user is permitted to append the new word to the existing word base. This may be simply done thru an interface when the token not found in the projected word base. This is done to append a legal word to the word base according to the lexical category, say if verb, verb template is used for storing the item. However the user has to give explicitly the root and its morphemes manually as the lexicon demands the knowledge about the lexical array.

For e.g. the word 'divide' is to be appended , the user has to pick up the verb option so that the verb template is displayed in order to enable the user to fill up the lexical array.

:VERB ENSE_FORMS NOMINAL_DERIVATIONS PLURAL_FORMS ADJECTIVE ADVERBIAL_FORMS PREFIXED_FORMS:

As per this template the lexical array corresponding to the verb pattern, the word may be entered as follows.

divide s d ing -sion

The notions for entry are very simple as substitute if the projection calculus is understood. The lexical semantics or fixation of syntactic markers is not discussed in the projection arithmetic. However the analysis phase elaborates the modified lexicon with the decimal markers along with the lexical fields to indicate the type syntactic category of the word.

5. Projection Mathematics

The 'Head' word is to be affixed with its successive tokens to get its full forms as described earlier. The array containing all the tokens is projected as the full forms of the Head Word (HW) to enhance the searching operation. For this purpose the following rules are adapted. There are 483 words are tabled and around 3000 words are associated with them. The projection rules table gives the complete manipulation of entries in the lexical array of the word base in order to expand the lexicon for the above said analysis. The following is the table.

Table 1: Projection Rules

1. *The Head Word, Head Word + affix, if no special symbols are attached to the affix.*
2. *The Head Word only, if no affixes.*
3. *The Head Word, if affixes with special symbols, triggering second level process.*
   - *a) if affix is preceded by a - symbol of the form -{affix}, the last character of the HW is to be truncated during projection.*
   - *b) if the affix is preceded by a number and - symbol of the form n-{affix}, then 'n' number of characters are to be truncated.*
   - *c) if the affix field has ~ symbol, the symbol is to be replaced by the HW.*
   - *d) if the affix is followed by the ~ symbol like {affix}~, the symbol is replaced by the HW along with the affix (prefix).*
   - *e) if the affix is followed by ~ and = as {affix}~=, the field will replace the HW (as this is a derived form of HW). All the rules are again applicable to the new HW during projection.*
   - *f) * replaces the previous field of the affix array.*
   - *g) # in the affix field indicates that the present field is a derived form of the HW and need*

not be affixed to HW and it is independent (to be projected as it is).

h) *$ represents the new HW and is to be replaced by the same during projection. This occurs for special case only.*

i) *If HW carries ^ symbol ({HW}^{alphabet}), it implies that the last character of HW is to be replaced by its following character before projection.*

j) *if the affix happens to be +, it represents tenses of the verb;*

The projection will follow the following pattern.

s, ing, and ed will be suffixed to the HW.

k) *if the affix is -+ , the rule will project by suffixing the tense endings viz s, -ing, and d to the HW.*

if the affix is e+, the substitution will be ed ing and ed.

l) *! tells the routine that the following fields are not to be affixed to the HW.*

m) *\ in the affix denotes that the particular word is to be projected as it is without affixing.*

The notion for the derivational features is given in the form of few suffices which is exclusive for this word base only. For example the following suffices would substitute the corresponding notions in the lexicon during projection. The notions are in upper case for distinction as the lexical projection is made case sensitive. However other analyses take the sentences as lower case words for uniform processing.

Table 2 : Projection Management for Suffices

Notion	Suffix	Sample Lexical Field	Projection
A -	*age*	*Break A*	*Breakage*
AL -	*al*	*logic AL*	*logical*
AY -	*ability*	*Match AY*	*match ability*
AI -	*ability*	*common AI*	*Commonality*
B -	*Able*	*Manage B*	*Manageable*
ES -	*Ness*	*Large ES*	*Largeness*
F -	*Ification*	*justify -F*	*Justification*
FL -	*ful*	*hope FL*	*hopeful*
G -	*ing*	*gather G*	*Gathering*
I -	*ivity*	*act I*	*Activity*
L -	*ly*	*final L*	*Finally*
M -	*ment*	*manage M*	*management*
N -	*ion*	*Calculate N*	*Calculation*
R-	*ier*	*rectify -ier*	*Rectifier*
TN -	*ation*	*form TN*	*Formation*
V -	*ive*	*Act V*	*Active*

The prefix attachment to the head word is very easily handled as per the rule (e) like *mis~* will expand to ***miscalculate*** if the head word is ***calculate*** and *mis~=* would result in projection by making all the derivational patterns applicable to miscalculate also. This is same for other prefixes like un, in, pre, trans, re, be, etc.

6. Conclusion

It is not claimed here that this model word base is the universal lexicon for AP or NL systems. It is a quick and reliable model for our application which deals with NL statements for describing computer program in arithmetic operations and input and output commands only. This work, however, envisages the possibility of enhancing the model for wider application as it is very easy to do so. As every computational lexicon is supposed to be concise and space conscious, the designing of our word base is taken care of with terminology and corresponding morphology with the same consciousness. Another merit of the lexicon is the whole collection of words is pooled and concisely reduced very carefully to 91 headwords, and 483 solid words which could generate around 2000+

words during projection. The easy on-line appending of new lexical entries through the lexical template is another merit. For further research, updating lexicon or learning of new lexical items and solution to lexical ambiguity may be done by applying Neural or Cognitive computing methodologies. The semantic information and the incorporation of complete syntax categories and lexical semantics are being under research that is not in the paper.

APPENDICES

A1 : Data Collection

TOWARDS AUTO PROGRAMMING THROUGH NATURAL LANGUAGE

SAMPLE DATA COLLECTION SHEET - (QUESTIONNAIRE)

---

0. Name
1. Age
2. Address
3. Knowledge in Computers (in years)
4. Experience in Programming (n years)
5. Qualification
6. The following is the way of representing our requirements/specifications of the program through Natural Language.

"The input variables are BREADTH and HEIGHT whose values are 50.25 and 67.90 respectively. Multiply those variables and assign the result to variable. Finally display the result".

7. The following is the complete program (in BASIC) which suits to the above described specifications:

```
10 rem
20 input BREADTH, HEIGHT
30 a = breadth * width
40 print a
50 stop : end
```

8. The following is another BASIC program.

Now Imagine that you, Describe the working of the program to the computer or Specify how the Program should work or Explain the steps of the Program in order to solve the problem.

```
10 rem sample program
20 input a,b
30 sum = a + b
40 diff = a - b
50 mul = a * b
60 div = a / b
70 print sum, diff ,mul, div
80 stop : end
```

9. The description: (not more than 25 lines)

A2 : The Sample Lexicon array (partial)

```
calculate -+ -N *s -B -V : mis~=
call + ~s B : re~=
cancel s led led lTN *s
carry -yies -yied -yiG -yier *s
certain ty L un~
collect + N *s V *L : un~=
come s -+ came come : be=
command + *s M *s ing *L
common L ES AI : un~=
communicate -+ -N *s -V *L
declare -+ -TN *s
deficient -cy *L
define -+ -ition *s -B
equate -+ -N *s 2-I
equip s ped ped M *s
erase -+ r *s B
fit s ted ted ting *s B
form + TN ~ *s B : trans~= : re~=
free s ing d d ES
function + ~ *s B
```

fuzzy -i *L -iness

imitate -+ -N *s -V * *L

important -ce *L

increase s -+ d -ing *L -B

interrogate -+ -N *s -V * *L

justify^i es ~ing ed R *s B cTN

keep s ing kept

know s knew n ledge *ful * *L ing * *L ledgeB

large r st L ness

link + G *s ~ *s

load + ~ *s B

make s ing \made \made ~ *s B

manage + M *s r *s B

manifest s eing ed TN *s B

mask + ~ *s B AY : un~=

References

[1] Anick, Peter; Bergler, Sabine, Lexical Structures for linguistic inference; In Pustejorsky, J.; Bergler, S., (Eds), Proceedings on Lexical Semantics & Knowledge Representation, Lecture Notes in AI, Springer Verlog, p 121-135, 1992

[2] Bran Boguraev and Ted Briscoe, Computational Lexicography for NLP, Longman UK Limited, UK,1989.

[3] John Caroll and Claire Grover, The Derivation of a large Computational Lexicon for English from LDOCE, In Bran Bogurav and Ted Briscoe, Computational Lexicography for Natural Language Processing, Lonman, 1989.

[4] Filgeuiras, J; Damas, L; Moreira,N.; Tomas, A.P, Natural Language Processing, Lecture Notes in AI, Springer Verlog, 1991.

[5] Haas, Stephanie,W., Covering the vocabulary of technical abstracts using standard and specialized dictionary, Jrl. of Information Science, Vl. 8, 5, 1992, p 363-373.

[6] Igor Mel'cuk and Alain Polguere, A formal lexicon in the Meaning Text Theory, Jl. Computational Linguistics, 13, pp 261-275, 1987.

[7] Rinaldo, Frank, J.; Strutz, Robert, E.; Evens, Martin, W., Developing a Lexicon for Automatic KA; In Proceedings of the Conference on Expert System for Development, IEEE Press, p 74-78, 1994.

[8] Shalini Agrawal and Deepak Khemani(Supervisor), A multilingual Semantic Lexicon, M.S. Thesis, IIT, Madras, April, 1995.

[9] R.F. Simmons, Semantic Networks : Computation and Use for Understanding English Sentences, In Roger C. Schank and Kenneth M.Colby, (Ed), Computer Models of thought and Language, W.H.Freeman and Co., 1989.

[10] Sowa, John, F., Logical Structures in the lexicon, In Pustejorsky, J.; Bergler, S., (Eds), Proceedings on Lexical Semantics & Knowledge

Author's Biography :

Dr. Gopinath Ganapathy is the Professor and Head of the Dept of Computer Science, Bharathidasan University, India. He did his under graduation and post graduation in Computer Science and Applications in 1986 and 1988 respectively from Bharathidasan University, India. He obtained his PhD degree, in Computer Science in 1996, from Madurai Kamaraj University, India. Received Young Scientist Fellow Award for the year 1994 and eventually did the research work at IIT Madras. He published around 17 papers. He is the Life member of CSI, ISTE and member of ACM. He was a Consultant for a 8.5 years in the firms in the US and UK, including IBM, Lucent Technologies (Bell Labs) and Toyota. His research interests include Security, Patterns, NLP, Web Engg, and KDD.