Normalization Strategy of Logical Knowledge Representation for Text Document

Rabiah A. Kadir, T. M. T. Sembok, Fatimah Ahmad, and Azreen Azman

***Abstract*—This paper discussed the idea of the computer system capable of simulating understanding with respect to reading a text document. The research is concerned with the problem of generating sophisticated knowledge representation for the purpose of understanding the natural language. Due to that, a simplification form of logical-oriented model of knowledge representation called Pragmatic Skolem Clauses (PSC) is proposed to represent the semantic formalism for the computational linguistic. Each set of pragmatic skolem clauses containing at least one skolem constant, which shows the thematic role relationship between clauses. Semantically and pragmatically-accented approach will be discussed in this paper in the context of formal grammar and linguistic semantic.**

***Index Terms*—Semantic technology, logical method, knowledge representation, first order logic.**

I. Introduction

In this paper, the parsing algorithm used in implementing the simplification of logical form in knowledge representation will be discussed. The simplified form of logical model is a type of knowledge representation that is designed based on First Order Logic (FOL). The simplified form of logical-oriented model is known as Pragmatic Skolem Clauses (PSC) representation. To implement a parser, the grammar was written in a form called Definite-Clause Grammar (DCG). Each phrase structure (PS) rule is a clause for a predicate with two arguments, such as: S --> NP VP.

Knowledge representation is the symbolic representation aspects of some closed universe of discourse. The objective of knowledge representation is to make knowledge explicit. Knowledge can be shared less ambiguously in its explicit form and this becomes especially important when computer automation is applied to facilitate knowledge management. In knowledge management, to solve complex problems encountered through artificial intelligence, a large amount of knowledge and some mechanism for manipulating that knowledge to formulate solutions to new problems are needed. Knowledge representation is a multidisciplinary subject that applies theories and techniques from three other fields [1] – Logic, Ontology and Computation

Knowledge Representation can be defined as the application of logic and ontology to the task of constructing computable models of some domain [1]-[3]. Logic and Ontology provide the formalization mechanisms required to make expressive models easily sharable and computer aware. This means that the full potential of knowledge accumulation can be exploited. However, computers only play the role of powerful processors with different levels of richness in information sources. Logic representation has been accepted as a good entity for representing the meaning of natural language sentences [4], and allows more subtle semantic issues to be dealt with.

This paper divided into several sections. The following section will discuss on the related research on knowledge representation for natural language. Then the third section is concerned about computing the meaning representation of texts document to constitute of understanding. The text document translations build up the meaning representation and enforce syntactic and semantic agreements. The following section discusses the translation strategy into a simplified form of logical-linguistic to encode the syntactic and semantic aspect of each sentence in text document. Translators may be involved in a very wide range of activities outside the work of translation, ranging from involvement in the grammar and parsing technique, which plays a highly visible role in representing knowledge, to acting as computing or helping the further research such as query system, dialogue system or search engine purposes. Finally will be the conclusion of the work and the further research concerned.

II. Related Research

Natural languages are the ultimate knowledge representation languages that are used by everyone in communication. Aristotle began his study of knowledge representation with an analysis of the semantic categories and relationships expressed in natural language [1]. Natural language semantic is related to knowledge representation, which is a source of empirical data and also a source of rich formalisms and computable operations. Both stimulate and complement each other. Below are the traditional requirements for natural language representation [5], [6] in [7]:

Manuscript received November 13, 2012; revised December 31, 2012. This work is supported in part by the Ministry of Higher Education (MoHE), Malaysia under Grant LRGS/TD/2011/UITM/ICT/03.

Rabiah A. Kadir is with the School of Computer Science and Information System, Najran University, Kingdom of Saudi Arabia, Seconded from Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, 43400 UPM Serdang, Selangor, Malaysia (e-mail: rabiah.akadir@gmail.com).

T. M. T. Sembok is with the Kulliyyah of Information and Communication Technology, International Islamic University Malaysia, P. O. Box 10, 50728, Kuala Lumpur, Malaysia (e-mail: tmtsembok@gmail.com).

Fatimah Ahmad is with the Computer Science Department, Faculty of Defence Science and Technology, National Defence University of Malaysia (e-mail: fatimah@upnm.edu.my).

Azreen Azman is with the Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, 43400 UPM Serdang, Selangor, Malaysia (e-mail: azreen.azman@gmail.com).

1) Meta-language must be capable of representing precisely, formally and unambiguously any information presented by an inquiry.
2) Meta-language should facilitate the canonic translation from the syntax representation language.
3) It should facilitate subsequent application of reasoning in the course of the process of semantic analysis.

Knowledge representation is at the very core of a radical idea for understanding intelligence. Instead of trying to understand or build brains from the bottom up, its goal is to understand and build intelligent behavior from the top down, putting the focus on what an agent needs to know in order to behave intelligently, how this knowledge can be represented symbolically, and how automated reasoning procedures can make this knowledge available as needed [8]. A knowledge representation at the conceptual level can support inferences that are not possible at the level of character strings. The inference depends on the representation of linguistic expression for the question answering relation.

There is a research such as studied the problem of conjunctive query answering over acyclic description logic ontologies as knowledge representation has been done [9]. Two approaches of knowledge representation that actively in this domain of research are logical and ontology. However, this paper will be concentrated on logical-oriented model for represent the semantic knowledge representation and will be discussed further in the following sections.

III. Normalization Parsing Strategy

A practical parser should do more than just suggest whether or not a sentence is acceptable. It should also report the structure of the sentence. The parsing technique must be designed in a way that it communicates with the semantics precisely at the points at which the semantics begin to have the necessary information to provide helpful feedback [10] - [14]. For this purpose, we present the arguments for the assertion of incremental interpretation of natural language sentence by modeled bi-DCG parsing technique, based on DCG parser. This parser raises two steps and has been extended with the bi-clausifier functionality. The two steps represent a tree diagram that corresponds to Prolog structure and produces the representation itself. To illustrate our representing tree, consider the sentence as found in the passage entitled *Storybook Person Found Alive!*, with the sentence *winnie the pooh was written in 1925*, taken as an example. This same sentence can be equally derived as shown in derivation tree as indicated by Fig. 1.

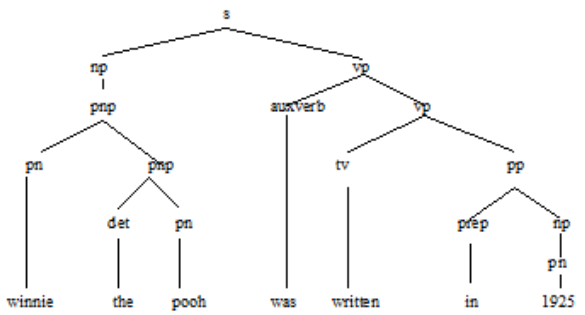

Fig. 1. Derivation Tree

The tree can be represented as the following phrase structure:

s(np(pn(winnie),pn(det(the),pn(pooh))),
vp(auxverb(was),vp(tv(written),pp(prep(in),
n(pn(1925)))))

To produce this representation, the parser will make each rule fill in the part of the structure which it is responsible for. For example, parsing begins with the rule s --> np, vp. This rule must therefore contribute the outermost s(..., ...) in the structure, where the portions represented by ... will be filled in by the np and vp rules, respectively. The np rule in turn will contribute np(..., ...) with arguments to be supplied by sub-rules within the above phrase structure.

Implementation of this process relies on the fact that the DCG notation allows extra arguments on predicates. If, for example, the following rule is written as:

s(a,b) --> np(c,d), vp(e,f). the translator will produce:
s(a,b,L1,L) :- np(c,d,L1,L2), vp(e,f,L2,L).

These extra arguments make a DCG more powerful than an ordinary phrase-structure grammar.

In the present case, the following arguments represent the tree. So the syntactic rules need to look like this:

```
s(s(NP,VP))            --> np(NP), vp(VP).
np(np(PNP))        --> pnp(PNP).
vp(vp(AUX,VP))     --> auxvev(AUX), vp(VP).
vp(vp(TV,PP))          --> tv(TV), pp(PP).
pnp(pnp(PN,PNP))   --> pn(PN), pnp(PNP).
pnp(pnp(D,PN))     --> d(D), pn(PN).
pp(pp(P,PN))       --> prep(P), pn(PN).
pn(pn(winnie))     --> [winnie].
pn(pn(pooh))       --> [pooh].
pn(pn(1925))       --> [1925].
d(d(the))              --> [the].
auxverb(auxverb(was)) --> [was].
tv(tv(written))        --> [written].
prep(prep(in))         --> [in].
```

When the first rule is invoked, its argument is immediately instantiated as s(NP,VP), but the variables NP and VP are not yet instantiated. The np rule then instantiates NP to np(PNP) so that the whole structure is s(np(PNP), VP) but PNP and VP do not yet have values. The structure will be completely instantiated when parsing is complete. Moreover, if execution backtracks out of a rule, the instantiations established by that rule are undone. The key idea here is that unification and instantiation gives way to working with information that do not yet have a value. This technique gives Prolog much of its power.

To solve the problem, the used of two parsing processes that proceed sequentially from the same input allow scanning of the input sentence in the same direction. This characteristic allows the use a normalize skolem constant for every single variable name in PSC representation. The lexicons together with the lexicon-dictionary are provided. The parser will use the DCG grammar.

The output of the first parsing is a collection of nouns with the skolem constant that will be used for the second parsing to generate the PSC representation. Each skolem constant was associated with the types of variable names. In this case, there are two symbols f_n represents the quantified variable names, while g_n represents ground term variable names.

IV. Translation Strategy

Translation rules are relatively simple because each of them is supposed to match the whole list of words. The output of a translation rule is a list of atoms which, when converted back into character strings and concatenated, will give the appropriate simplified form of logical-linguistic. The first of these rules handle the 'quit' command that the user will use to exit from the program. The procedure that applies the translation rule will simply find a rule that applies to the input, then execute a cut, or complain if no rule is applicable.

```
% translate (-InputSentence, +LogicalForm)
% Applies a translation rule, or complains
% if no translation rule matches the input.
parse( Sentence, LF, quit ) :-
                    quit(    LF, Sentence, [.] ).
parse( Sentence, LF, query ) :-
                    query(    LF, Sentence, [?] ).
parse( Sentence, LF, assertion ) :-
                    sentence( LF, nogap, Sentence, [.] ).
  translate(InputSentence, LogicalForm) :-
    parse(InputSentence, LogicalForm, Type),
    tr(LogicalForm, Clauses),
                    !.
  translate(_, []) :-
    write('I do not understand your sentence.'), nl.
```

To present the story passage into a simplified form of logical-linguistic, it is necessary to encode the syntactic and semantic aspect of each sentence. The parser recognizes two types of semantic entities: predicate and names, and its predicate arguments relation to give the relationship of these entities. It returns error message on receiving ill-formed input. An input is considered ill-formed if it contains one of the following conditions:

1) Unknown words – are words that are not predefined in lexicon, and these include misspelled words.
2) Non-covered lexicon-dictionary – the structure of the lexicons is not covered by the lexicon-dictionary implemented, even though it is grammatically correct.
3) Illegal grammatically syntactic structure – the structure of the input is grammatically wrong.

To describe the meaning of natural language utterances, a precised way of describing the information that they contained is needed. It relies on the logical model and set theory, both of which are precisely defined knowledge bases.

Consider a simple formula such as lives(chris, england) (*Chris lives in England*). This formula shows a part of a logical language. A logical model consists of an Entity (*E*), which is the set of individual people and things that can be talked about, plus a Semantic function (*S*) which gives a relation onto entities. This model has two important advantages. First, it assigns meaning to all parts of every formula, rather than just assigning truth values to a complete sentence. Second, a logical model works with knowledge bases without making any claims about the real world as a whole. This is important because it corresponds closely to computer manipulation of a database.

A. Logical Translation

Logic form is derived from the syntactic parse of the text input and each lexicon in the text will recognize two types of semantic entities: nouns and verbs. The first thing to be noted is that names are logical constant ('Chris' = chris), but common nouns, and noun with adjective are predicates ('children' = (λx) children (x)). An adjective, such as 'small' is considered a property, not an entity. This has to do with the distinction between sense and reference. A name refers to only one individual, thus the translation is directed to a logical constant. But a common noun such as 'children' can refer to many different individuals, so its translation is the property that these individuals share. The reference of 'children' in any particular utterance is the value of x that makes children(x) true.

Second, note that different verbs require different numbers of arguments. The intransitive verb 'barked' translates to a one-place predicate (λx) (barked(x). A transitive verb translates to a two-place predicate (λy) (λx) (cuts(x,y).

These arguments are filled in, step by step, as you progress up from common noun to NP, from verb to VP, and then S. The following example of text is used to serve an illustration:

- *"At noon, two small children cut a ribbon."*

noon(x1 ^ at(x1)) & two(x2 : (small(x2) & children(x2)) & exists(x3,ribbon(x3) & cuts(x2,x3))

- *"The ribbon was made from paper."*

exists(x4,ribbon(x4) & paper(x5 ^ makes(x4,x5))

B. Skolem Constant Generation

Before PSC can be generated, it is required to generate a new unique constant symbol known as Skolem Constant. Each logic expression involves predicate, functions and quantifier, so that the generation of skolem constant implements an algorithm to convert a formula into clausal form that has modified its skolem function. The following is an algorithm needed to convert logical formula into a logically equivalent sentence that is in a clause form [15].

1) Eliminate all connection (<=>) by replacing each instance of the form ((P <=> Q) by the equivalent expression ((P => Q) ^ (Q => P)).
2) Eliminate all connection (=>) by replacing each instance of the form (P => Q) by (~P ∨ Q).
3) Reduce the scope of each negation symbol to a single predicate by applying equivalents such as converting:
   a) ~~P to P
   b) ~(P ∨ Q) to ~P ^ ~Q
   c) ~(P ^ Q) to ~P ∨ ~Q
   d) ~(Ax) P to (Ex) ~P
   e) ~(Ex) P to (Ax) ~P
1) Standardize variables - rename all variables so that each quantifier has its own unique variable name. For example, convert (Ax)P(x) to (Ay)P(y) if there is another place where variable x is already used.
2) Skolemizing - eliminate existential and universal quantification and ground term by introducing Skolem functions. For example:
   a) (Ex)P(x) to P(c) where c is a brand new constant symbol that is not used in any other sentence. c is called a Skolem constant.
   b) More generally, if the existential quantifier is within the scope of a universally quantified variable, then introduce a Skolem function that depends on the universally quantified variable. (Ax)(Ey)P(x, y) is

converted to (Ax)P(x,f(x)). f is called Skolem function, and must be a brand new function name that does not occur in any other sentence in the entire knowledge bases.

c) P(x) to P(c) where c is a brand new constant symbol that is not used in any other sentence. c is called a Skolem constant.

1) Remove universal quantification symbols by first moving them all to the left end and make the scope of each the entire sentence, and then just drop the 'prefix'. For example, convert (Ax)P(x) to P(x).
2) Distribute "and" over "or" to get a conjunction of disjunctions called conjunctive normal form. Convert:
   a) (P ^ Q) ∨ R to (P ∨ R) ^ (Q ∨ R)
   b) (P ∨ Q) ∨ R to (P ∨ Q ∨ R)
3) Split each conjunction into separate clauses, which is just a disjunction ("or") of negated and un-negated predicate, called literals.
4) Standardize variables apart again so that each clause contains variable names that do not occur in any other clause.

For this first parsing, the transformed formula and the list of variables have been introduced by universal and existential quantifier, and ground term. Skolem function makes use of two new predicates. Predicate gensym must be defined such that the goal gensym(X,Y) causes Y to be instantiated to a new atom built up from the atom X and a number. This is used to generate skolem constant that have not been used before. The second new predicate mentioned is subst. Here it is required for subst(V1, V2, F1, F2) to be true if the result of substituting V2 for V1 every time it appears in the formula F1 is F2.

```
skolem(all(X,P), all(X,P1),  Vars) :- !,
    skolem( P, P1, [X|Vars] ).
skolem(exists(X,P), P2, Vars) :- !,
    gensym( f, F ),
    Sk =..[F|Vars],
    subst( X, Sk, P, P1 ),
    skolem( P1, P2, Vars ).
skolem(Pred(X:P), Pred(F)&P2, Vars) :- !,
    gensym( g, F ), Sk =..[F|Vars],
    subst( X, Sk, P, P1 ),
    skolem( P1, P2, Vars ).
skolem((P & Q), (P1 & Q1), Vars ) :- !,
    skolem( P, P1, Vars ),
    skolem( Q, Q1, Vars ).
skolem((P # Q), (P1 # Q1), Vars ) :- !,
    skolem( P, P1, Vars ),
    skolem( Q, Q1, Vars ).
skolem(P, P, Vars).
subst(X, Sk, exists(Y,P), exists(Y,P1)) :- !,
    subst( X, Sk, P, P1 ).
subst(X, Sk, (P & Q),(P1 & Q1)) :- !,
    subst( X, Sk, P, P1 ),
    subst( X, Sk, Q, Q1 ).
subst(X, Sk, P, P1) :- functor(P,F,N),
gensym(Root, Atom) :-
     get_num(Root, Num),
     name(Root, Name1),
     integer_name(Num, Name2),
     append(Name1, Name2, Name),
     name(Atom, Name).
get_num(Root, Num) :-
     retract(current_num(Root, Num1)), !,
        Num is Num1+1,
        asserta(current_num(Root, Num)).
get_num(Root,1):-
    asserta(current_num(Root, 1)).
```

In the process of transformation, the normalization of the skolem constants are applied to all variable names. We identified two types of skolem constant to differentiate between quantified (f_n) and ground term (g_n) variable names. The following shows the use of f_n and g_n which stand for skolem constant in clausal form for each variable names.

```
cls(two, g9).
cls(small, g9).
cls(children, g9).
cls([ribbon, f55).
cls([paper, g10).
cls(pretty, f3).
cls(home, f3).
cls(three, g4).
cls(old, g4).
cls(year, g4).
cls(poem, f4).
```

Each skolem constant that are generated will be stored in the list of normalization clauses skolem constant for the second parsing process.

C. Final Parsing

Based on the research problem, before the resolution theorem prover can be applied, a set of simplified formula is required to be converted into what is known as clausal form. This section explains the process of transforming the simplified logical formula into clausal form, called PSC. This transformation is a second parsing, whereas the step is the same as the first parsing which implemented an algorithm to convert a simplified logical formula into clausal form. However, since the skolem function has been modified, instead of generating a new skolem constant symbol, it will retrieve an atom that was already built up in the first parsing.

```
skolem(Pred(X:P), Pred(F)&P2, Vars ) :- !,
          getatom( Pred, F ),
          Sk =..[F|Vars],
     subst( X, Sk, P, P1 ),
     skolem_v2( P1, P2, Vars ).
getatom(Noun, Atom) :-
        (cls(Noun, Const) ->
         (name(Const, ListTemp),
         name(Atom, ListTemp))
          ;
         gensym_v2(g, Atom)).
```

The following shows a set of PSC as knowledge base representation that can be applied in the context of natural language question answering system. For example, after the transformation process, we will have the following representation is created.

```
two(g9)
small(g9)
children(g9)
ribbon(f55)
paper(g10)
```

cuts(g9,f55)
makes(f55,g10)
pretty(f3)
home(f3)
calls(f3,r(cotchfield & farm))
lives(chris,f3)
three(g4)
old(g4)
year(g4)
isa(chris,g4)
poem(f4)
about(f4,him)
writes(r(mr & robin),f4)

V. Discussion

The PSC capability is unified a standard constant clause pragmatically for a text document. However, the process still relies on the fact that the DCG notation allows extra arguments on predicates. The implementations of two parsing processes that proceed sequentially from the same input allow scanning of the input sentence in the same direction. This characteristic allows the use a normalize skolem constant for every single variable name in PSC representation that able to give the pragmatic relationship for the whole of text document. This proposed logical form of knowledge representation may cause the question answering will be able to extract the relevant answers.

VI. Conclusion

Text documents are directly translated into logical representation form which can be used as a complete content indicator of a query system. The translation technique used has been described in this paper in the earlier sections. The text documents are processed to form their respective indexes through the translation and normalization process which are composed of simplification processes. This representation is used to define implication rules for any particular question answering system and for defining synonym and hypernym words.

For further research, the query is translated into its logical representation as documents are translated. The representation is then simplified and partially reduced. The resulting representation of the query is then ready to be proven with the document representation and their literal answers are retrieved. The proving will perform through uncertain implication process where predicates are matched and propagated, which finally gives a literal answer value between the query and the document.

Acknowledgment

This work was supported by the Ministry of Higher Education (MoHE), Malaysia via National University of Malaysia under Long Term Research Grant Scheme LRGS/TD/2011/UITM/ICT/03.

References

[1] J. F. Sowa, *Knowledge Representation: Logical, Philosophical and Computational Foundation*. Pacific Grove, USA: Thomson Learning, 2000.

[2] R. Zajac, "Towards ontological question answering," in *Proc. Asso-ciation for Computational Linguistics, Workshop on Open-Domain QuestionAnswering*, pp. 31-37, 2001.

[3] L. T. Lim and E. K. Tang, "Building an ontology-based multilingual lexicon for word sense disambiguation in machine translation," in *Proc. of the 5th Workshop on Multilingual Lexical Databases*, Papillon, 2004.

[4] I. Bratko, *Prolog Programming for Artificial Intelligence*, Great Britain: Addison-Wesley, pp. 555-580, 2001.

[5] W. A. Woods, "Network grammars for natural language analysis," in *Reading in NLP*, California: Morgan Kaufman, 1979.

[6] D. H. D. Warren and F. C. N. Pereira, "An efficient easily adaptable system for interpreting natural language queries," *American Journal of Computational Linguistics*, vol. 8, no. 3-4, 1982.

[7] B. Galitsky, *Natural Language Question Answering System*, 2nd ed.. Adelaide: Advanced Knowledge International Pty Ltd., 2003.

[8] R. J. Brachman and H. J. Levesque, *Knowledge Representation and Reasoning*, San Francisco, CA: Morgan Kaufmann, 2004.

[9] B. C. Grau, I. Horrocks, M. Kroetsch, C.Kupke, D. Magka, B. Motik, and Z. Wang, "Acyclicity conditions and their application to query answering in description logics," in *Proc. of the 13th International Conference on Principles of Knowledge Representation and Reasoning (KR 2012)*, 2012.

[10] E. P. Giachin and C. Rullent, "Robust parsing of severely corrupted spoken utterances," in *Proc. COLING-88*, pp. 196-201, 1988.

[11] E. K. Tang and A. H. Mosleh, "Example-based natural language parsing based on the SSTC annotation schema," in *Proc. of National Conference on Research and Development in Computer Science and its Applications*, pp. 1-6, 1997.

[12] A. H. Mosleh and E. K. Tang, "A flexible example-based parser based on the SSTC" in *Proc. of 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pp. 687-693, 1998.

[13] S. Zhu, M. Zhou, X. Liu, and C. Huang, "An efficient stochastic context-free parsing algorithm," *Journal of software,* vol. 9, no. 8, pp. 592-597, 1998.

[14] Y. Zaharin and C. Boitet, "Representation trees and string-tree correspondences," in *Proc. of Coling-88*, pp. 59-64, 1988.

[15] W. F. Clocksin and C. S. Mellish, *Programming in Prolog*. Heidelberg, Berlin: Springer-Verlag., 2003.

Rabiah A. Kadir was borned in East Malaysia, Sarawak year 1969. She enrolled her diploma in Computer Science in 1987 after finishing her schooling at College Science Datuk Patinggi Abang Haji Abdillah, Kuching Sarawak, Malaysia. Rabiah furthered her study in first degree of Computer Science year 1990 at Universiti Pertanian Malaysia. In year 1997, she graduated her Masters in Computer Science at Universiti Kebangsaan Malaysia. After several years, she enrolled her PhD in Computer Science with a major field in computational linguistic on December 2003 and completed her study on May 2007. She graduated her PhD from Universiti Kebangsaan Malaysia. During her study in Masters and PhD, she was attached with Universiti Putra Malaysia as a tutor and lecturer respectively. She is a senior lecturer in Department of Computer Science, Faculty of Computer Science and Information Technology, Universiti Putra Malaysia with a specialization in computational linguistics in Artificial Intelligence Research Group. Currently, she is seconded with Najran Universiti, Kingdom of Saudi Arabia as an Assistant Professor. Her research interests include information retrieval and expert system. Rabiah Abdul Kadir joins Malaysian Information Technology Society (MITs) since year 2008 as Vice Treasurer. She is also a member to the Malaysian Information Retrieval and Knowledge Management Society. She was awarded two gold medals for her PhD research work in BIS and Eureka Exhibition in year 2007. Currently, she had published more than 30 journals and international proceedings.

Tengku Mohd Tengku Sembok has over thirty years of experience in various fields of Information Communication Technology. He has taught undergraduate and postgraduate programs and managed numerous R&D and consultancy projects successfully. He had supervised 30 PhD students successfully to completion. He obtained his B.Sc.(Hons) in Computer Science from Brighton Polytechnic in 1977, MS from Iowa University in

1981, and PhD from Glasgow University in 1989. His last appointment was Deputy Vice Chancellor (Academic and International Affairs) in the National Defence University of Malaysia. He currently holds a chair of senior professor in Computer Science at International Islamic University of Malaysia. He has held several academic posts at UKM prior to his current assignment. His current research areas are in computational linguistics (for Malay, English and Arabic languages), artificial intelligence, information systems, information retrieval, multimedia courseware, language technology, and knowledge management. He has published over 100 articles in these areas. He has also received numerous awards in international invention and innovation exhibitions for his research products in Geneve, Brussels, and London. He was awarded the national ICT Excellent Teacher 2004 jointly by Ministry of Science, Technology and Innovation, Ministry of Energy, Water and Communication, Malaysian National Computer Confederation, and MAXIS Bhd. He is a Fellow of Academy of Sciences Malaysia, and a fellow of Malaysian Science Association. Currently he chairs the Engineering and Computer Science Discipline of the Academy of Sciences Malaysia. He is also the Chairman of Society of Information Retrieval and Knowledgement Malaysia.

Fatimah Dato Ahmad is a Professor at the Department of Computer Science, Faculty of Defence Science and Technology, National Defence University of Malaysia (NDUM). Currently, she is the Director of the Centre for Information and Communication Technology, NDUM. She obtained her Ph.D. from the National University of Malaysia in 1995. Her research interests include information retrieval, multimedia computing, and natural language processing. Prof. Dr. Hjh. Fatimah Dato Ahmad is a member of the Malaysian Information Technology Society (MITS), the Institute of Electrical and Electronics Engineers (IEEE) and also the Malaysian Information Retrieval and Knowledge Management Society (PECAMP) where she is a life member. Currently, she has published more than 100 articles in journals and proceedings both local and international.

Azreen Azman was born in Negeri Sembilan, Malaysia in 1977. He received a Diploma in Software Engineering from the Institute of Telecommunication and Information Technology in 1997. Immediately, he was accepted directly to second year in Multimedia University, Malaysia to study Bachelor of Information Technology majoring in Information Systems Engineering. He completed his bachelor degree in 1999. After serving in the industry for a few years, he enrolled for a Ph.D in January 2003, studying Computing Science specializing in Information Retrieval in the University of Glasgow, Scotland and completed his study in September 2007. After completing his bachelor degree, Azreen Azman joined Motorola Semiconductor in Seremban, Malaysia on July 1999. The company was later changed to ON Semiconductor where he served as System Analyst until December 2002 before pursuing his Ph.D study. He was also briefly employed by ON Semiconductor as System Engineer upon returning from his Ph.D study in 2007. On January 2008, he joined Universiti Sains Islam Malaysia as a lecturer and head of programme for Bachelor in Computer Science majoring in Information Security and Assurance. He later joined Universiti Putra Malaysia in May 2009 as a senior lecturer in the Department of Multimedia, Faculty of Computer Science and Information Technology. He is also a member of Digital Information Computation and Retrieval research group. His research interest are information retrieval and text mining. Azreen Azman joins Malaysian Society of Information Retrieval and Knowledge Management since 2008 and recently serve as Committee Member. He is also a member of Malaysian Information Technology Society. Currently he also serves Malaysian Qualification Agency as panel of assessors for programme accreditation.