From 'single parent' to 'bound pairs': the secret life of computerese

Keith Tong Sai-tao

Introduction

This discussion takes an initial look at the use of selected lexical items in Computer Science text through corpus-building and concordancing techniques, and attempts to propose a prototype entry for the possible compilation of a glossary of technical and sub-technical terms in this discipline. The work branches from a Research Grants Council-funded research project jointly conducted by the Language Centre, HKUST and the Department of English, City Polytechnic of Hong Kong, in collaboration with the Department of English, Guangzhou Institute of Foreign Languages, entitled An empirical investigation into dimensions of difficulty in comprehension of the academic English of Science and Technology.

Principal objectives of the RGC-funded project

The project, which focuses on comprehension as an indisputable need in Science and Technology courses at tertiary level, has two principal objectives:

(a) a determination of the nature of Hong Kong tertiary students' comprehension problems in reading and listening in English for academic purposes; and

(b) the development of teaching and learning materials to enhance reading and listening skills, informed by the findings of the empirical enquiries.

The present paper deals only with the reading component of the project.
The reading component

Probably something between 60% and 80% (depending on the discipline) of all research material in Science and Technology worldwide is published in English. Rapid and effective comprehension of such reference material is thus vitally important for students at tertiary level. However, students who approach this task as second-language users of English often encounter difficulties in comprehension in terms of both lexis and syntax. With the former, the biggest problem seems to be the inability to identify the particular sense of a word used in context. Research conducted in the UK (e.g. Wang 1988) has shown that when Chinese students use a dictionary as an aid in reading, they often accept the first definition given in the dictionary entry, totally oblivious of the variety of senses that a word may have and also of the fact that words take on special meanings in different contexts. The reading component of the present study seeks to harness computer technology to investigate the linguistic features of Science/Technology writing in English, in order to isolate areas of potential difficulty and to generate materials to help students overcome problems in these areas.

Building a corpus of Computer Science text

It was originally envisaged that a corpus of one million words of Science and Technology text could be built, which could then be divided into subsets of various disciplines. When it was later realised that a one-million-word corpus was unrealistically small given the vastness of the domain, it was decided that a corpus of text in one specific discipline should be compiled instead. Computer Science was chosen because it is the only subject within this domain taught at all of the tertiary institutions in Hong Kong.

Sampling

Determination of the range of samples was difficult, for the areas of Computer Science are difficult to delimit. First-year reading lists were obtained from the Computer Science departments of each of the tertiary institutions. It was discovered that some of the texts that Computer Science majors are asked to read belong to peripheral subjects such as Psychology, Management Information Systems, Economics and General Business. A decision had to be made, therefore,
to include only texts which were more specifically ‘Computer Science’ in orientation. The chosen texts cover topics such as algorithms, networking, databases, programming languages, artificial intelligence and expert systems. After the books had been selected, a random sampling procedure was carried out to extract approximately 6,000 words from each. Each book was divided into three equal sections, and then a random number was generated for each section, from which approximately 2,000 words were selected. Since sentences are kept intact, the number of words taken from each section usually slightly exceeds 2,000. (Computer language/pseudocode and mathematical expressions/equations are cut from the running text and are replaced by the protocols *CL* and *EQ* respectively.) The product of this sampling and extraction process is a total number of 1,000,842 tokens, comprising 26,934 types, extracted from 166 titles (the HKUST Corpus). See James et al. (1992); James (in press).

**Analysis of corpus data**

Now that the corpus compilation is complete, analysis is being carried out into the characteristics of the texts sampled. The focus of this paper is on aspects of lexical use in such texts, with concentration on the use of ‘sub-technical’ terms.

A glance at the word-frequency list for the corpus shows that function words such as the, of, a, to, is and and top the table, which is not surprising as no matter how technical the writing is, it is still written English prose. The first content words that appear are common technical terms in computer contexts, such as data, program, system and function. This is hardly any more surprising, and probably not particularly interesting, since intuition would have predicted these as the high frequency words, and arguably Computer Science majors would have no problems understanding such key terms. A more interesting area to explore is probably that of sub-technical vocabulary.

**Sub-technical vocabulary**

Trimble (1985:129) defines sub-technical vocabulary as:

> Those words that have one or more ‘general’ English meanings and which in technical contexts take on extended meanings (technical, or specialised in some fashion).
One example Trimble gives is *fast*, a general enough word, but meaning ‘resistant to’ in Medicine and ‘a hard stratum under poorly consolidated ground’ in Mining, as well as referring to paints whose colours are “not affected by light, heat, damp”.

By the same token, there are bound to be such ‘general’ words which take on specialised meanings in the context of Computer Science writing. They are not the straightforward technical terms such as *disk-drive* or *database*, and probably do not occur nearly as frequently. But they will have to be used frequently and widely enough to warrant attention, surely not words with a frequency count of less than 20. It was intuitively thought that words in the region of, say, 50 to 100 in frequency counts are those to which attention should be paid. The picking of general English words which may take on specialised meanings was done by educated guesswork. It was decided that initial investigations were to be made of the four items: *parent, child, bound* and *hold*.

**Parent**

According to the *Encyclopedia of Computer Science and Engineering* (ECSE 1983), the word *parent* is used in two major technical senses in the realm of Computer Science. The first such sense concerns graph theory:

> ... rooted trees in computer literature are drawn upside-down, with the root at the “top”. Computer terminology for tree structures is usually sexist. Every root is said to be a father (mother) to the roots of its subtrees. These roots are called “brothers” (sisters) and “sons” (daughters) of their “fathers” (mothers). The nearer words (parent, child, sibling) are sometimes used. The root of the entire tree (godhead?) has no father. The non-sexist words “ancestor” and “descendant” are terms that may designate nodes several levels apart in the tree. (ECSE 1983:666)

The second major technical sense of *parent* pertains to the topic of ‘computer memory’. A quotation from the chapter on ‘Memory’ in the *Encyclopedia* goes:

> ... it is often desirable in commercial data processing (and certain scientific applications) to continually create backup copies of tape master files .... The old master file is called the “parent”, the new master file the “child”; when the next updating is performed, a new generation of this file is created; the “parent” file becomes the “grandparent“, the “child” becomes
the "parent", etc. Often, 30 or more generations are kept for vital corporate master files; e.g., daily updatings for a month. (ECSE 1983:957-958)

It is apparent that in both these two technical uses the word *parent*, and for that matter, the word *child*, are used **metaphorically**. In graph theory, it depicts how one node ‘gives birth to’ another or how a node ‘descends’ from another. In describing the generation of backup files, it refers to the original file, from which backup files are ‘born’. No doubt all non-native users of English who are pursuing their studies in Computer Science through the medium of English would have acquired a certain level of English at which the most common meanings of *parent* and *child* would not pose a problem. However, one has less confidence in assuming that all of them will be able to make the necessary leap of imagination to appreciate fully the metaphorical use. The situation seems to warrant attention by the ESP teacher in the classroom as well as the highlighting of useful information in instruments such as specialist dictionaries and glossaries. The concordanced data from our corpus sheds light on how the word *parent* is used in Computer Science text. Table 9.1 below shows the number of occurrences of the lemma *parent*, which includes citations such as *parent*, *parental*, *parentage*, *parents* and *parent-child*.

<table>
<thead>
<tr>
<th></th>
<th>No. of occurrences</th>
<th>Technical use</th>
<th>Non-technical use</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>parent etc.</em></td>
<td>82 (23)</td>
<td>74 (21)</td>
<td>8 (5)</td>
</tr>
</tbody>
</table>

Table 9.1: Number of occurrences of the lemma *parent* in the HKUST Corpus. (The numbers in brackets refer to the numbers of extracts in which the items appear.)

The citations of *parent* occur 82 times in the total of 1,000,842 tokens, and are found in 23 out of 166 titles. These figures are too large to be ignored. Interestingly, in 74 out of the 82 instances, the use is a technical one. The eight uses of the non-technical *parent* are from only five titles, showing that such use is sporadic, and does not recur in a small local context. A closer look at the concordanced samples reveals that the technical use of *parent* is often recurrent in a localised context. For example, the citation of *parent* is found to occur 14 times in one extract of one title.

A breakdown of the information about individual citations yields some interesting observations. See Table 9.2 below.
<table>
<thead>
<tr>
<th></th>
<th>No. of occurrences</th>
<th>Technical use</th>
<th>Non-technical use</th>
</tr>
</thead>
<tbody>
<tr>
<td>parent</td>
<td>66 (17)</td>
<td>61 (16)</td>
<td>5 (3)</td>
</tr>
<tr>
<td>parental</td>
<td>1 (1)</td>
<td>1 (1)</td>
<td>0</td>
</tr>
<tr>
<td>parentage</td>
<td>1 (1)</td>
<td>1 (1)</td>
<td>0</td>
</tr>
<tr>
<td>parents</td>
<td>7 (7)</td>
<td>4 (4)</td>
<td>3 (3)</td>
</tr>
<tr>
<td>parent-child</td>
<td>7 (3)</td>
<td>7 (3)</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 9.2: Number of occurrences of the different citations of parent in the HKUST Corpus. (The numbers in brackets refer to the number of extracts in which the items appear.)

One salient feature is the disparity between the citation of parent and that of parents. When used in its most common meanings one would expect parents to be used much more frequently than parent. But when used metaphorically in technical writing, occurrences of parent far outnumber those of parents. There are perhaps two main reasons for this. Firstly, in the human sense, it takes two parents to conceive a child; in Computer Science, each parent, whether meaning a node in a tree diagram or a master file, is singular in nature. The second reason is the prevalence of noun compounds in writings in Science and Technology. Noun compounds can be defined as:

two or more nouns plus necessary adjectives ... that together make up a single concept; that is the total expresses a ‘single noun’ idea.

(Trimble 1985:130-131)

The concordanced samples of parent show that this item is often used to modify other nouns in such compounds. The most frequently found collocations include parent data flowchart (8 counts), parent-child relationship (8 counts), parent node (6 counts), parent record type (5 counts), parent (dataflow) diagram (4 counts).

It is also interesting to note that of the five instances where the citation of parent is used in the general non-technical sense, two are used metaphorically in noun compounds, as the examples below show:

1. **This can happen, as when a parent company pays a bill for a subsidiary whose name is quite different and pays the wrong amount.**

2. **As with the parent field of information studies itself, the topic of information systems is a rather chaotic hybrid field.**
Apparently the function of *parent* in expressions such as *parent company* and *parent field* is not unlike that in expressions such as *parent data flowchart* and *parent record type*. Indeed, the term *parent company* is treated as a distinct item in some general dictionaries (e.g. LDOCE1, LDOCE2). However, the two technical expressions *parent data flowchart* and *parent record type* remain so and are unlikely to find their way into the general dictionary. The question remains as to whether the Computer Science major who has to read texts in a second language needs proper induction by the teacher and/or learning materials to derive the metaphorical meanings from the expressions. The answer is probably "yes", as it is worth reminding the student that in Computer Science text *parent* more often than not takes on a special technical meaning which is metaphorical in nature. Noun compounds where *parent* plays a modifying role are perhaps not too difficult to handle, but it is when the student encounters sentences like the following that ambiguity may strike if the student fails to appreciate the technical meaning:

3. *It is based on the principle that a “single” parent can have “many” children and a child can have only “one” parent.*

**Child**

The use of *child* in a technical sense is found to resemble very closely that of *parent*. This is hardly surprising because *parent* and *child* are two complementary notions in Computer Science use. Table 9.3 below shows the number of occurrences of *child* and *children* in technical and non-technical uses.

<table>
<thead>
<tr>
<th></th>
<th>No. of occurrences</th>
<th>Technical use</th>
<th>Non-technical use</th>
</tr>
</thead>
<tbody>
<tr>
<td>child</td>
<td>86 (12)</td>
<td>61 (10)</td>
<td>5 (3)</td>
</tr>
<tr>
<td>children</td>
<td>35 (15)</td>
<td>25 (8)</td>
<td>10 (7)</td>
</tr>
</tbody>
</table>

Table 9.3: Number of occurrences of *child* and *children* in the HKUST Corpus. (The numbers in brackets refer to the numbers of extracts in which the items occur.)

The number of occurrences of *child* in technical use is much greater than that of *child* in non-technical use. There are only four occurrences in which *child* is used in the general sense. (The total number of occurrences of the technical use of *child* is greatly boosted by the heavily recurrent use (25 instances) of the word in one
The most common collocations found are *left child* (12 counts), *right child* (12 counts), *child data (flow)chart(s)* (11 counts), *child (record) types* (5 counts) and *child directory/directories* (4 counts). Below are four concordanced samples:

4. One contains the address of the *left child*, and the other the address of the *right child* (See Figure 10).

5. Therefore, the data storage must be shown on both the parent and *child data flowcharts*.

6. *In other words*, for any given occurrence of any given parent record type, there are *n* occurrences of each of its *child record types* (*n* greater than or equal to zero).

7. *Any child directory has exactly one parent directory and none or more child directories.*

The use of *children* is interesting. First of all, it occurs much less frequently than *child* (35 instances vs. 86). When *children* in technical use is compared with *child* in technical use, the contrast is even greater (25 instances vs. 81). This can be partly explained by the fact that *child* (but not *children*) is very often used as a modifier in noun compounds such as *child data flowcharts, child record types* and *child directories*. Further, *children* is used in a general sense in rather varied contexts, as the following examples show:

8. *In the near term, a salient question may be how to take advantage of the interests and cognitive skills that young children may be coming to school with as the result of experiences with computers in the home or learned in the first years of school.*

9. *Without these jobs, most of the mothers would have been required either to get remote jobs and lose a lot of contact with their children, or to go on the welfare rolls.*

10. *Women, children, personal computers – no one is safe.*

11. *For each person, you want to be able to retrieve their date and place of birth, date and place of death, and the names and birth dates of each of their children.*
12. *Summer camps are run for ten-year-old children to teach them BASIC, LOGO, and other languages.*

13. *Some would say this is an instance of the old proverb that holds that the shoemaker's children are always the last to get new shoes.*

14. *He lives with his wife and children in California.*

However, it is quite unlikely that students will fail to distinguish between *children* in technical use and *children* in general use. It will nevertheless help them if information about such metaphorical uses of *child* and *children*, and for that matter, *parent* and *parents*, can be made available to them.

*Bound*

The word *bound* has two major, distinct uses in Computer Science. It can be expressed as the past participle form of the verb *bind*, and thus functions as an adjective, meaning 'fixed', as opposed to 'free'. The concept of *binding* is very important in programming. One definition of *binding* is:

the assignment of directly machine-computable values to the symbolic expressions of which a source program is generally composed.

(ECSE 1983:890)

Thus, *bound values. Bound* is used in a similar sense in lambda calculus, which is a topic in functional programming. The *Encyclopedia* explanation goes:

The lambda calculus (or λ-calculus) is a mathematical formalism developed by the logician Alonzo Church in the 1930s to model the mathematical notion of substitution of values for bound variables. Consider the definition \( f(x) = x + 1 \), which defines \( f \) to be the successor function. The variable \( x \) in this definition is a *bound variable* in the sense that replacement of all instances of \( x \) by some other variable (say, \( y \)) yields a definition \( f(y) = y + 1 \), which is semantically equivalent.

(ECSE 1983:822)

The second major technical sense of *bound* in Computer Science pertains to its use as a noun to mean 'limit', mainly in the area of memory protection in programming. In this respect, usually two limits are set, namely the *upper bound* and the *lower bound*. 
Information from the HKUST Corpus reveals how frequently *bound* is used in each of these two technical senses and how often in more general ones. See Table 9.4 below.

<table>
<thead>
<tr>
<th>Technical use of bound</th>
<th>Non-technical use of bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>adjectival use</td>
<td>nominal use</td>
</tr>
<tr>
<td>45 (11)</td>
<td>43 (15)</td>
</tr>
</tbody>
</table>

Table 9.4: The different uses of *bound* in the HKUST Corpus.
(The numbers in brackets refer to the numbers of extracts in which the items occur.)

One interesting observation which can be made is that the nominal use of *bound* to mean 'limit' does not occur in non-technical contexts. However, it occurs nearly as often as the adjectival use in technical contexts, and in fact, *slightly more widely* than the latter, as it is found in 15, as opposed to 11, of the 166 titles. A search through three of the most commonly used general dictionaries, namely COBUILD (1987), LDOCE2 (1987) and OALD4 (1989), reveals no inclusion of this particular sense of the word. On the other hand, although this use is spread out over 15 different titles, there are clusters in which the occurrences are densely represented in a small local context. For example, 12 occurrences are found in the Extract 3 of Title 037, while another 12 are found in the three Extracts of Title 056. The most frequently found collocations of *bound* in this use are, not surprisingly, *upper bound* and *lower bound*. Below are concordanced samples illustrating this use.

15. *If the constant expression specifying the bound is missing, the array has an incomplete type.*

16. *The definition of a subrange simply indicates the least and the largest constant value in the subrange, where the lower bound must not be greater than the upper bound.*

17. *Out-of-bound array references cannot, in general, be detected at compile-time.*

18. *The most obvious way to do this would be 'EQ' but this is no good by itself because it makes the weights grow without bound, and one unit comes to dominate the competition for all inputs.*
The uses of *bound* in (17) and (18) are particularly interesting, in that they seem to be attempts at marrying lexical items in specialised technical use and common idiomatic use of English, resulting in hybrids which may be readily comprehensible to the technically-informed but probably not immediately recognisable to the layman. The expression *without bound* (18) cannot be found in general dictionaries of English, though *without limit* can. The hyphenated modifier *out-of-bound* (17) is not in common use either, though the expression *out of bounds* (with a different meaning) is. The questions are whether these expressions are emerging as idioms in ‘computerese’ as they become more commonly used, and whether the Computer Science student will recognise their restricted use in Computer Science contexts.

The adjectival use of *bound* in general non-technical contexts is recorded in eight occurrences in eight different Titles, indicating the commonness of this use and at the same time justifying the classification of *bound* as ‘sub-technical vocabulary’. On the other hand, the adjectival use of *bound* in technical contexts to mean ‘fixed’ is recorded in 45 occurrences in 11 Titles. Of these occurrences, 19 are clustered in two Extracts of Title 054, while the others are rather thinly spread out over the remaining ten Titles. A search through the concordanced samples reveals one of the most common collocations for this use of *bound*: with the preposition *to* and followed by a noun (phrase). Examples of such use are as follows:

19. *The value of the tail is bound to the variable Tail and passed on to each level of recursion until the boundary condition is met.*

20. *When located, the variable Value will be bound to the fourth element in the list.*

*Bound* is also frequently used to modify nouns such as *variable(s)*, *value(s)* and *pair(s)*. Below are some examples.

21. *A bound variable is treated exactly the same as the object it is bound to.*

22. *The bound values themselves must belong to an ordinal type (real is proscribed), and are often supplied as user-defined constants.*

23. *Bound pairs are separated by two dots.*
The use of *bound* is apparently common and varied enough to warrant attention by the language teacher, so that students of Computer Science will acquire a degree of awareness about the various uses, especially the two major technical uses. This is especially important when sometimes *bound* (adj.) and *bound* (n.) exist in the same sentence, as is shown below:

24. *It has a LET* form with parameters *bound* to old and new upper and lower bounds: *'CL'*. 

25. *Once the LET* parameters are *bound*, each proposed bound is compared with the existing bound, and if it is different, the new bound is placed in the appropriate slots.*

**Hold**

*Hold* is a word which can be said to be very ‘general’ in that it is a frequently used verb with a wide range of senses, and forms numerous idioms when combined with other words. It will be interesting to discover whether there is any particular sense of *hold* which stands out as most commonly used in Computer Science contexts. An examination of the concordanced data of the three citations of *hold*, *holds* and *held* yields several observations.

Firstly, there is one particular technical sense which is heavily represented in the total number of occurrences, namely, ‘to store information in the computer system’. Table 9.5 below shows the frequencies in which *hold* is used to mean ‘to store (information)*.

<table>
<thead>
<tr>
<th></th>
<th>No of occurrences</th>
<th>Technical sense: 'to store (information)'</th>
<th>Other senses</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>hold</em></td>
<td>121</td>
<td>72</td>
<td>49</td>
</tr>
<tr>
<td><em>holds</em></td>
<td>76</td>
<td>31</td>
<td>45</td>
</tr>
<tr>
<td><em>held</em></td>
<td>74</td>
<td>48</td>
<td>26</td>
</tr>
<tr>
<td><em>hold + holds + held</em></td>
<td>271</td>
<td>151</td>
<td>120</td>
</tr>
</tbody>
</table>

Table 9.5: Numbers of occurrences of three citations of *hold* to mean ‘to store (information)’ in the HKUST Corpus.
It can be seen from Table 9.5 that in more than half of their occurrences *hold*, *holds* and *held* are used to refer to the ‘storing of information in computer systems’. Below are some examples from concordanced samples.

26. *Of course, the buffer cannot hold all of a program’s modules and screens at once.*

27. *For a situation such as the above, a dynamic storage allocation strategy would grab enough memory to hold 400 double values, then 200, then 100, and so on.*

28. *To resolve this problem, we will define a record variable called CurrentMaster that holds the account information of the Master file record that is currently processed.*

29. *A tree comprises nodes, each of which holds some data and pointers to two or more descendant nodes;*

30. *They are held in memory as fixed-length fields.*

31. *data store: a reservoir in which data can be held for an indefinite period.*

Another observation is that the intransitive use of *hold* to mean ‘(for a proposition) to be valid, to stand, or to apply’ is surprisingly common in Computer Science text. Table 9.6 below provides some information about the distribution of this use among the three citations of *hold*, *holds* and *held*.

<table>
<thead>
<tr>
<th></th>
<th>Total no. of occurrences</th>
<th>Intransitive use to mean ‘to stand, to be valid, to apply’</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>hold</em></td>
<td>121 (73)</td>
<td>17 (11)</td>
</tr>
<tr>
<td><em>holds</em></td>
<td>76 (47)</td>
<td>30 (18)</td>
</tr>
<tr>
<td><em>held</em></td>
<td>74 (36)</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 9.6: The intransitive use of *hold*, *holds* and *held* to mean ‘to stand, to be valid, to apply’ in the HKUST Corpus. (The numbers in brackets refer to the numbers of extracts in which the items occur.)
It is apparent that this use is quite widely spread over all of the titles, as is the use of all three citations in all senses. The zero occurrence for held is not surprising, as this use is intransitive in nature, precluding the possibility of passive voice. One feature which reveals itself in the concordanced samples is that hold is often used in this way in sentences stating conditions. Below are some examples of the intransitive use of hold to mean ‘to stand, to be valid, to apply’, the last three illustrating its use in sentences of condition.

32. ... if the message has not been delivered but the sender knows this fact, then the message is not lost and the following two propositions hold.

33. The run time part of the system includes a run time checker, to test that assumptions upon which applications were precompiled still hold at run time;

34. The post-condition specifies a condition which must hold after the operation is applied.

35. The fundamental fact that must hold true for a topographically organized system is that nearby units respond similarly.

36. Compound statements allow you to specify a group of statements to be performed if a condition holds or while a condition holds.

37. This result holds for values of H up to about 40 in the case of XOR.

38. The same relationship holds regardless of the cell to which the formula is copied.

This use of hold is thought to be difficult for the non-native user of English for two reasons. Firstly, it is unlikely to be found in general English textbooks for learners of English, but would rather be found in formal academic writing. Secondly, this use of hold comes somewhat low in the list of definitions for hold (v.) in a general English dictionary and the chances are that most students would not have the perseverance to press on long enough to locate this particular sense. It is, therefore, thought desirable to include it in the ESP teaching and learning materials designed for Computer Science students.
A third observation about the use of *hold* is that the wide range of its senses is rather strongly represented even in Computer Science text. Apart from the senses discussed above, 16 others are identified in the concordanced samples of the three citations of *hold*, *holds* and *held*, viz.

(i) v.t. to control with fingers

  e.g. *Hold* the disk as shown in Figure 2.

(ii) v.t. to keep, to retain

  e.g. *Data processing people tend to follow technology, so your old systems won’t hold them on the job unless they like the security.*

(iii) v.t. to defend, keep intact

  e.g. *It helps hold the line on costs without jeopardizing your competitive stance.*

(iv) v.t. to maintain (in a certain condition)

  e.g. *Clearly we need something above these structure charts to hold them together.*

(v) v.t. to assert

  e.g. *The optimists hold that the computer-assisted manufacturing trends described in this chapter and the resulting outlook for productivity gains will lead to a higher standard of living, a shorter workweek, and increased leisure time.*

(vi) v.t. to have (as an opinion, belief, view, etc.)

  e.g. *The remaining things are grouped according to concepts and perceptions we hold about what it means to be “like”.*

(vii) n. control

  e.g. *As low-level design, coding, test plans, and publications plans are being developed, a disturbing realization takes hold.*
(viii) v.t. to maintain in a position/at a level

   e.g.  For this chip, simply hold ME low and WE high, and select the desired address.

(ix)  v.t. to support

   e.g.  Lockout at a low level, the field level for instance, doesn't hold up the rest of the database, but is a comparatively high-overhead solution.

(x)   v.t. to cause to fulfil an agreement or promise; to bind

   e.g.  Furthermore, it is not necessary or advisable to hold developers to the preliminary estimates included in the project plan.

(xi)  v.t. to have; to possess

   e.g.  Today people can hold jobs based on interest and level of skills rather than on whether they are male or female.

(xii) n. temporary inaction

   e.g.  Eleanor then puts the call on hold and uses an intercom to announce the name of the person being called and the number of the line on which the caller is waiting.

(xiii) v.t. to regard

   e.g.  Why do we hold interpretation of the environment to be so important?

(xiv) v.t. to conduct

   e.g.  As the central technical resource, the leader holds regular interactions with each member of the technical team: KEs, software engineers, and other support personnel (both technical and administrative) who may be assigned to the task.
(xv) v.i. to maintain one's grip

e.g. *A job typically acquires tokens of a passive queue and holds on to them while visiting other queues active and/or passive queues* and model elements.

(xvi) v.t. to take up; to assume (responsibility)

e.g. *The responsibility for understanding the entire statement, in depth, together with all of the assumptions and conceptualizations on which it rests, is typically held by no one.*

### An annotated glossary

The foregoing analysis seems to justify the categorisation of words into sub-technical vocabulary, members of which possess the following characteristics. They are general English words which are very commonly found in non-technical writing. However, they take on specialised technical meanings in technical writing. In such technical contexts, their use in the specialised technical sense is dominant, greatly outnumbering their uses in general senses. The words discussed above, namely *parent, child, bound* and *hold*, all seem to possess these characteristics. The non-native student of English encountering such vocabulary in technical use may not be able to deduce the contextual meaning immediately. The keen student may turn to general English dictionaries or specialist dictionaries for help. A quick search through the two most commonly used general English dictionaries used by tertiary students in Hong Kong (I.DOCE2 and OALD4) reveals that the exercise may not be very helpful. The technical meanings of the four words in Computer Science text are not entered in either dictionary. There is an entry for *hold* to mean 'to contain', which is close enough to 'to store computer information'. The general meaning of *hold* to mean 'to stand or to continue to be valid' is entered in both dictionaries. This is as much help as the student can get.

A cursory search was also made of three specialist dictionaries to see whether the sub-technical vocabulary items in question can be found. They are the *Penguin Dictionary of Computers* (PDC 1985), the *Dictionary of Computer Terms* (DCT 1989) and the *English Chinese Dictionary of Microcomputer* (ECDM 1989). Table 9.7 below summarises the findings.
<table>
<thead>
<tr>
<th></th>
<th>PDC</th>
<th>DCT</th>
<th>ECDM</th>
</tr>
</thead>
<tbody>
<tr>
<td>parent</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>child</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>bound</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>hold</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 9.7: The inclusion of four sub-technical vocabulary items in three specialist dictionaries.

None of the four items is found in DCT. The sole entry for *hold* in PDC gives the following explanation:

To preserve data in one *storage device* after transferring it to another *storage device* or another *location* in the same device. Contrasted with *clear*.

This explanation seems to be inadequate, as, from the data provided by our corpus, this is only one meaning, and *hold* can also mean ‘contain’ or ‘accommodate’ in certain other technical contexts.

Only one meaning is given for *bound* in the entry in ECDM, that of ‘limit’, but the meaning of *bound* (adj.) to mean ‘fixed’ is missing. Again, only one meaning is given for *hold*, that of ‘to keep in a certain state or level’, but the more common uses of *hold* as evidenced by our corpus data are not included.

Apparently, the editors of specialist dictionaries have chosen to include mainly technical vocabulary but have neglected sub-technical vocabulary. Now that a million-word corpus of Computer Science text has been constructed, the data can be used to compile a corpus-based dictionary or glossary which includes both technical and sub-technical vocabulary. The decisions on inclusion of lexical items can be made systematically with the help of frequency lists and careful examination of concordanced data. It is suggested that an entry in such a dictionary or glossary should have the following characteristics:

(a) The various definitions for one entry should be sequenced according to frequency of occurrence.

(b) The number of occurrences and the number of extracts in which the occurrences appear should be given so that the reader will know the commonness of the use and the width of representation.
(c) Apart from an explanation of the meaning and use, examples of typical use should be provided from concordanced data.

(d) Common collocations should also be suggested. Again, statistics could be provided about the commonness of such collocations.

**Conclusion**

The findings of this study seem to justify the case for examining the sub-technical vocabulary used in a particular technical subject. Data from such investigations will shed light on the lexical features of the writings in the subject, and help compile useful instruments for studies, such as specialist glossaries. However, in the long run, empirical data obtained about the text must be tested against the actual users of such texts, namely, Computer Science students, to see whether the perceived difficulties really exist or not. Such findings will certainly help in the design of teaching and learning materials for Technical English courses.