A Critical Learning Skill

Ronald I. Frank
rfrank@pace.edu
IS Dept. Pace University,
Pleasantville, NY 10570

Abstract

This is a "heads-up" about an important (critical) learning skill we all need to learn about, teach, and develop. The skill is to be able to distinguish cognate word meanings. Technical terms derived from ordinary English have specific and different meanings from their ordinary antecedents. First, students have to be explicitly made aware of this phenomenon. Second, special attention has to be applied to isolating technical meanings when and wherever they occur. Third, repetition of the cognate meaning is the key to instilling the new meaning in the students' vocabulary. We give some examples of terms having this characteristic, followed by classroom experience and an final summary.

Key Words: Learning Skills, Pedagogy, Semantics

1. INTRODUCTION

The problem.

Ordinary words are sometimes usurped for highly specialized technical use. The learning skill is to recognize the specialized, often unique, technical meaning stipulated by the writer or speaker. The problem is well documented for mathematics and discussed in detail in (Edwards & Ward 2004). It is my observation that this is also a problem for computing. Being a new field, we introduce terms at a high rate, often by bending the meaning of existing words to a new use.

Cognates are words in different languages that derive from a common root (OED, 2005). This is exactly what often happens in computing. In (Edwards & Ward 2004), students were not aware that word definitions (what they understand) can be either "extracted" from previous experience or "stipulated" in the subject at hand. Being unaware of this difference, the students fell back on their natural extracted word understanding and missed the stipulated meanings, thereby missing the technical concepts being taught Our responsibility is to recognize that we too have this problem and therefore we need to focus on its remediation. We need to teach students to be aware of the cognate problem, to adjust for the difference, and to learn correct meanings. The learning skill is to be able to distinguish (filter) words read or heard so that the correct technical meaning and ideas are recognized.

Some people deny the existence of this problem. The fact that many texts go to the trouble of isolating word lists for each chapter, is a very direct proof that technical terminology is a central learning issue in computing. See, for example, (Dennis, 2005).

Outline of this note.

To better understand the learning skill, we first define the problem and its solution. Then we'll build a useful analogy and apply it to filtering technical readings and lectures for better understanding. We then discuss how to develop this learning skill, and show some examples of where it is needed. There is a short summary of skill building at the end, and a bibliography.

As instructors we have to explicitly teach both the fact of stipulated meanings and then the critical learning skill and how to develop it.

Some people think this is merely the problem of knowing about contexts, and that meanings change with contexts. It is, and it is more. When in a specific technical context, the precision of definition is often very much higher than it is in common experience. This is from the very nature of science and technology, which draw precise distinctions in order to isolate and analyze phenomena. Also, who is to say that we naturally and consciously learn about contexts and consciously learn to evaluate meanings relative to the current context? It seems from the cited study (Edwards & Ward 2004), that this is NOT a skill that either comes naturally nor is it usually taught. It is probably true that we all do adjust to context to some extent, in an unconscious untutored way. The point of this article is that the mental adjustment has to be tutored and conscious and practiced.

Anyone who has studied modern computer languages such as C++, C#, and Java is aware of the concept of "overloading". A function or method name can be reused (redefined) in a program (or object) so long as the argument list in each definition occurrence is unique in its number of arguments and or the type sequence of its arguments. This allows the compiler to disambiguate function calls. The point of this article is that our mental adjustment that allows us to dynamically disambiguate has to be tutored and conscious (and practiced).

2 THE SOLUTION

1. First we must consciously understand and realize (internalize) that there is a difference in some words (terminology) between what we know and what we have to learn in a new field. This can be taught by examples such as those below.

2. We must focus on the stipulated meanings in the field under study and learn these new meanings. This is often aided by making notes in a notebook for review of definitions and procedures (if the words describe a process). This is a neglected language learning skill itself.

3. We have to develop the skill of recognizing the context being discussed and therefore which meaning of a term we should use. How do you get to Carnegie Hall (the learning skill)? Practice, practice, practice. How do you learn a new term? Practice it. I.e., use it at least three times in context.

2 A USEFUL ANALOGY

A useful analogy is filtering light through a polarized lens (Larimore, 1965, or Wikipedia) as in Polaroid sunglasses. Polarized lenses let through light with a given linear polarization, say up-down. They filter out (block) light with horizontal (left-right) polarization. This takes out the glare from water, glass, metallic painted cars, and other non-bare metal, horizontally polarizing surfaces, making it easier to see them. In some cases, it changes a blindingly bright scene into one that can be seen and appreciated. The blindingly bright light comes from the light energy being concentrated in the horizontal mode by surfaces that polarize the light. The trick to understanding polarized light and lenses is to realize that light can simultaneously contain light waves oscillating in ALL directions (circularly polarized light). The blindingly bright scene mentioned above contains mostly blinding horizontal light AND some useful vertical light. When the Polaroid lens filters out most of the horizontal light component, what's left is the clearly seen mostly vertically polarized view that can be appreciated.

3. APPLYING THE ANALOGY

Applying this analogy to words in our brain is direct. Our brain "sees" words in that the incoming words (written or aural) carry meaning, which is sensed in our brain. What many people fail to realize is that words, like light, can simultaneously carry many meanings. Some meanings are at or near to what was intended, and other meanings are completely orthogonal (independent of) to what was intended. Just like circularly polarized light some of the correct meaning of words can get through without filtering (along with a lot of confusing garbage leading to some but poor or confused understanding).

In technical work, the "light" (word meaning) is often highly polarized in one direction. If we let in even a little of the wrongly directed light (other meanings) we miss the intended meaning completely.

4 DEVELOPING THE LEARNING SKILL

How do we develop this filtering skill? First, by fully realizing that words simultaneously carry multiple meanings. Next, by noting that in most cases there is one useful technical meaning that is intended in the context at hand, and that most other meanings are either not closely related, or if closely related, they can differ by a significant nuance. This is analogous to light that is polarized far away from the vertical, or if nearer to the vertical, is still significantly off the vertical.

This situation of confounding the meaning of words is particularly bad in computing. Computing is a fast-changing field that requires new terms at a high rate of introduction. The easiest way to coin a new term is to take an old one and refit it to a new meaning. Usually a term is chosen and its use modified slightly. It then is misused by the press or colleagues who mutate its meaning until it can take on a radically different meaning.

The way to build the required critical learning skill (filter) so that it is available to apply in real-time, is to carefully note down those terms that are defined at variance to well known "common usage", and then review the list once in a while. Writing using the new terms helps a great deal. By explicitly realizing the variant meaning when we come across it, we tell our brain to note it. When we write it down and then review it, we reinforce the new meaning in our head. A well known principle of learning is based on The Bellman's Rule of Three, (Carroll, 2008): using it three or more times will solidify it in our filter.

> "Just the place for a Snark! I have said it twice:
> That alone should encourage the crew.
> Just the place for a Snark! I have said it thrice:
> What I tell you three times is true."

5 SOME EXAMPLES

Effort

In IS project management we deal with the effort to do a project. We all know what effort is. Not so. The COCOMO (Boehm, 2000) and related models of information systems development projects use the term "effort" in estimating project cost and duration. Effort is defined ONLY and precisely as person-months spent, and nothing else.

Object

In object oriented analysis and design an object is a thing. Some noun or pronoun naming a thing. True but only slightly relevant as a learning tool. An object (Dennis, 2005) is an instance of an encapsulation of data and the code needed to manipulate the data. This kind of an object can be used to represent the thing of interest - but it is not the thing itself.

Process

In operating systems a process is, informally, a more or less clearly defined procedure for doing something. This does not help us understand operating systems. A process (Stallings, 2005) is a single thread of executable code and ALL of the resources (such as CPU time, files, main memory space, and CPU registers) needed for its execution, and their important values (the state). This specific kind of process can execute the first mentioned kind.

Agile

In software development, to be agile is to be quick-moving, nimble, or active (OED, 2005), reacting to the changing environment. Again, an analogous but not precise meaning. The agile manifesto (Ambler 2008) provides explicit criteria to judge whether or not a project is being done in an agile manner (four guidelines and 12 principles).

Group

In discrete math we use a group of things. Sort of a set of similar things. Yes, but not helpful. Every element of a group possesses a very specific set of properties (Gallian, 2002). If a thing fails to have

even one of the properties, it is NOT a member of the group. A group is exactly:

a. *A set of elements.*
b. *A binary operation (*) combining two elements to give a third: d = a*b.*
c. *The operation is associative: a*(b*c) = (a*b)*c.*
d. *There is a special element (called the identity) e such that a*e = e*a = a for ALL elements a in the group*
e. *For each element a there is an element b (called the inverse of a) such that a*b = b*a = e.*

The trick is to remember EXACTLY what the group properties are. IF you do, you will get the old joke that YOU too can join the group and associate with it only if you bring your own inverse

Interface

In Object Oriented Programming (Java and UML in particular) we use the term "interface" (Flanagan, 1999). Everyone should know the general meaning of the term and its use to describe the graphics mode of computer use. The OOP meaning includes the general meaning: the protocol for communication between two system components. The OOP meaning has some very specific additional meanings. So many that to just quote the general meaning of interface is to miss the critical difference. The protocol of the interface is the set of callable variables and methods. The interface (the set of variables and methods) has a stipulated name. If Class A is to provide the interface, it must:

a. *Have a statement in a class (A) that it provides the named interface.*
b. *Provide the declarations in A of all callable elements of the interface - its variables and methods.*
c. *Provide the definitions in A of all callable elements of the interface - its variables and methods.*

This major critical difference is that the author of class A must create the details of the named interface (write the set of variables and methods) and place them inside class A.

Declaration/Definition

In example 6 above, we used two words in a highly precise technical meaning that may have slipped by unappreciated. We referred to both the declarations and the definitions of the interface elements. To declare a variable, for example, is to mention it in a stipulated syntax that causes memory to be assigned to hold it. However, to define it is to actually state the value to be held in memory.

Package

Package is another object-oriented term (Dennis, 2005) that is confusing. An OO package *is* a package, in the vernacular sense, but it is also used in a limited technical sense, especially in Java (Flanagan, 1999). It is most often used to refer to a construct which contains closely related classes (a library) that perform a specific set of tasks. For example, java.io or java.lang.

Equals

"Equals" is a widely used term. In math we distinguish = from ≡ (identity). The first glyph means "has equal value", as in x = 2. The second glyph means "always has equal value" as in x ≡ (2x-x), for all possible x values.

In programming, we have different ways things can be equal. Two data items stored in different locations can have the same (=) value. Two data items, referred to by different names can be equal because their names actually refer to the same memory location. This is a stronger form of being equal. It is analogous to the math meaning of identity.

So what do we mean when we program "A = B" ? Actually neither! We mean take the value of B and assign it (store it into) A. Whether we store the actual data item or only store B's address in A is up to the programming language and syntax we are using. If we store the data then A and B are different but equal in value. If we store B's address in A, they are identical since both names refer to the same stored data item.

If at a later time we want to query their equality, we could say (A == B) which is a test of their values and is true or false. In the second case above it yields false. The values of the memory data differ. One is a

value and one is an address. In this case, the syntax gives us a smarter test. We could say (A.equals(B)), which tests this case properly and returns true only if they both refer to the same stored item.

Parameter

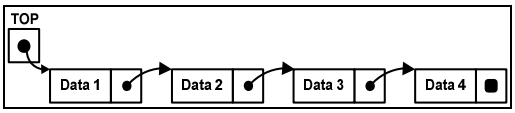
Parameter is an overloaded word, meaning that is has multiple technical meanings. A parameter is a constant that can change. It does not change as often or as much as a variable, but it can change. A constant never changes. There are religious wars over when a parameter is a variable or a constant.

On the other hand, a parameter is an element in a list that is sent into a program through its calling sequence. The input parameter can be a constant, a variable, or a parameter (first meaning). So, what am I saying when I say that "the first parameter is the variable x, the second called parameter is a constant c, and the third is a parameter of the process"? (Flanagan, 1999).

List

The word "list" used in 10 above is another example. The meaning there is of a sequential set of symbols, which are usually names set off by commas - something like a shopping list. However, list also has a technical meaning. It is a data structure wherein each piece of data is paired with an address field containing the address of the next list item.

There are also two special list items. The first, (called TOP) has no data and is just the address of the actual first data/address list item. The second is actually the very last list item. It has a data field but no valid address in its address field. Instead, it contains a marker that indicates to list processing programs that they have processed to the end of the list.



What is interesting and confusing is that the simple list - in the parameter list - is often stored in a - list data structure for further processing.

6 CLASSROOM EXPERIENCE

I lecture on this problem and in each course I emphasize the most problematic terms. I urge students to take noted that include tech-term meanings. The words then appear on tests. All of the test words are reviewed afterwards in class to further practice their meanings. Some students still have a continuing difficulty mastering technical word meanings.

7 SUMMARY

There are unique useful technical meanings to some ordinary words. We have to explicitly teach this. We should teach students to consciously notice them and develop the habit of writing a word list down when the terms are first encountered. Word lists in books should be emphasized more than is customary in courses.

We should teach the habit of reviewing the personal list and using each term at least three times.

7 REFERENCES

Ambler, Scott, et al. http://www.agilemanifesto.org/ Last accessed 1/4/8.

Boehm, B., et. al. Software Cost Estimation With COCOCMO II. Pp.392 on. Prentice Hall PTR, Upper Saddle River, NJ. 2000. ISBN 0-13-026692-2

Carroll, Lewis, The Hunting of the Snark: The Bellman's Rule of Three, Fit the First, The Landing. http://etext.virginia.edu/etcbin/toccer-new2?id=CarSnar.sgm&images=images/modeng&data=/texts/english/modeng/parsed&tag=public&part=1&division=div1 Last accessed 1/4/8.

Dennis, A., Wixom, B., and Tegarden, D., Systems Analysis and Design with UML 2.0. 2nd Ed. Wiley, Reading, MA. 2005. ISBN 978-0-471-34806-1

Edwards, B. S., & Ward, M. B. "Surprises from Mathematics Education Research: Student (Mis)use of Mathematical Definitions". MAA Monthly, Vol. 111, pp. 411-424, May 2004.

Flanagan, D. Java In A Nutshell 3rd Ed. Pp. 61 on. O'Reilly. Sebastopol, CA. 1999. ISBN 1-56592-487-8.

Gallian, J. A. Contemporary Abstract Algebra 5th Ed. P. 43. Houghton Mifflin, NY. 2002. ISBN 0-618-12214-1

Larmore, Lewis. Introduction to Photographic Principles, Pp. 76-81. Dover Pubs. NY 1965. Lib. of Congress 65-20484.

OED. The Shorter Oxford English Dictionary Sixth Ed. Oxford University Press USA. (2002 - 2007). CD ROM Version 3. (2005) ISBN 978-0-19-923176-8.

Stallings, W. Operating Systems 5th Rd. Pg. 66. Pearson Prentice Hall, Upper Saddle River, NJ. 2005. ISBN 0-13-147954-7

Wikipedia. http://en.wikipedia.org/wiki/Polarization Last accessed 1/4/8.