A Critical Learning Skill

Ronald I. Frank
Ronald I. Frank

Educator
Pace University – Full time Associate Professor IS Dept. (1998 - present)
- Java, Telecommunications, Software Development, Operating Systems, and Project Management
- DPS Doctoral Program dissertation advisor.

Pace University – Adjunct while at IBM Research Center (1985 - 1998)
- CS Telecom, Software Development, Operating systems, and Programming Languages (C, Fortran, Pascal)

Massachusetts College System – Adjunct while at IBM Cambridge Scientific Center (1974 - 1984)
- All courses in CS & IS & Telecom.

Information Technology Researcher and Innovator, Senior R & D Project Manager, International Consultant IBM (1960-1993)
- Research Division staff member, manager & corporate-board consultant
- Principal IBM US spokesman/consultant for the engineering and scientific PC
- Scientific Center staff member / project manager and staff member & computer center manger
- District and branch offices award winning Scientific Marketing Rep.
- Service Bureau Corp. development programmer: commercial & scientific applications

IBM Projects
1. Corporate-board consultant [educational uses of the internet]
2. Research manager of parallel computer interfaces
3. Research manager of low-end APL (for workstations)
4. Research staff member (APL language and Virtual Reality research)
   - Team produced the first multi-venue VR experience
5. Team created the first Remote Operation of [370/3090 Main Frame] Computers
   - VM OS Programmable Operator (PROP)” component.
6. Team created first modern LAN Network Management.
7. Team developed the first WAN line quality monitoring technology.
8. APL Development team - Earliest “Home Use” Of Personal Computers
   - Developed Scientific and Engineering applications for “proto personal computers”.
9. Scientific Marketing Consultant to ~10% of US Industrial Research. (Esso (sic), ATT, Bell Labs, General Dynamics Electric Boat, etc.)
   - Special Scientific rep to the University of Pennsylvania / Consultant to Provost on developing a computing facility for Academic/Scientific use
   - IBM Education Centers: Created some of the earliest courses and taught IBMers and customers what computers were and how they could be used.
10. Manager IBM Systems Research Institute Computing Center [A grad School & research center]
11. Developed the first computer molecular model at the IBM NYSR&D Center (SHARE contribution).

- A virtual reality R & D company. Chairman and CEO.

Education
Pace University CSIS DPS in Computing. (2002)


Rutgers University: BA, Math (1960)

Hobbies
I like to hike the Westchester North County Trailway, snorkel, documentary photography, and read science fiction - not all at the same time.
This is a heads-up about an important (critical) learning skill we all need to know about and develop.

The learning skill.

The learning skill is to be able to distinguish (filter) words you read or hear. You filter them so that the correct technical meaning and ideas get through to your brain. You distinguish the precise technical meaning from all others. Most people naively think we just read or hear and understand. Not so.

Outline of this note.

To better understand the learning skill, I'll first define the problem and its solution. Then I'll build a useful analogy and apply it to filtering technical readings and lectures for better understanding. I'll 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.

The critical learning skill of the title is the ability to dynamically (in real-time) filter what you read or hear so that you receive the intended technical meaning. It has been shown recently [1] that this skill is missing in many math students leading to their poor showing in math classes. We can reasonable assume the same missing learning skill in most novices in ANY technical field, and in fact even in many experienced technically oriented folks including journalists.

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 as discussed in [1] is that 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. We need to be aware of the difference, to know how to adjust for the difference, and how to learn the correct meanings. This is the critical learning skill we need to develop.

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 [1], that this is NOT a skill that either comes naturally or is 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.

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 the function calls. The point of this article is that our mental adjustment that allows us to dynamically disambiguate has to be tutored and conscious.

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.

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.

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.

A useful analogy.

A useful analogy is filtering light through a polarized lens [2, 3] 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.
A Critical Learning Skill

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.

Applying the analogy to the learning skill.

Applying this analogy to words and your brain is very direct. Your brain "sees" words in that the incoming words (written or aural) carry meaning which is sensed in your 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.

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 gets 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. By explicitly realizing the variant meaning when you come across it, you
tell your brain to note it. When you write it down and then review it, you reinforce the new meaning in your head. A well known principle of learning is based on The Bellman's Rule of Three [4]: using it three or more times will solidify it in your filter.

Some examples:

1. In IS project management we deal with the effort to do a project. We all know what effort is. Not so. The COCOMO [5] 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.

2. 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 [6] 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.

3. 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 [7] 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 kind.

4. In software development, to be agile is to be quick-moving, nimble, or active [8], reacting to the changing environment. Again, an analogous but not precise meaning. The agile manifesto [9] provides explicit criteria to judge whether or not a project is being done in an agile manner (four guidelines and 12 principles).

5. 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 [10]. 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.

6. In Object Oriented Programming (Java and UML in particular) we use the term "interface" [11]. 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.

7. 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.

8. "Speed" and "Velocity" are often misused. We talk of the velocity of light as a universal constant (in vacuum). Actually it is its speed that is constant. Its velocity is its speed in a given direction. Clearly, the direction of a beam of light is variable. Only its speed is not (in vacuum). Speed is a scalar. Velocity is a vector (scalar length + direction) [12].
9. "Equals" is a widely used term. In math we distinguish \( = \) from \( \equiv \). The first glyph means "has equal value", as in \( x = 2 \). The second glyph means "always has equal value" as in \( x \equiv (2x-x) \), 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 \( \equiv \).

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 \( (A \equiv B) \) 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.

10. 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 (meaning 1). 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? [13].

11. 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.

Summary:

1. There are unique useful technical meanings to some ordinary words.
2. Consciously notice them.
3. Note them down when first encountered.
4. Review the list.
5. It also helps to use each one at least three times.
6. Expect to see them on tests and in the rest of your career.
A Critical Learning Skill

© Dr. Ronald I. Frank 2008
325 Goldstein Academic Center
Pace Pleasantville

E-mail: RFrank@Pace.edu

Bibliography:


"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."


[12] Any textbook on basic physics or vector analysis.

The Ivan G. Seidenberg
School of Computer Science and Information Systems
Pace University

Technical Report Series

EDITORIAL BOARD

Editor:
Allen Stix, Computer Science, Pace--Westchester

Associate Editors:
Constance A. Knapp, Information Systems, Pace--Westchester
Susan M. Merritt, Dean, The Ivan G. Seidenberg School of Computer Science and Information Systems --Pace

Members:
Howard S. Blum, Computer Science, Pace--New York
Mary F. Courtney, Computer Science, Pace--Westchester
Nicholas J. De Lillo, Mathematics and Computer Science, Manhattan College
Daniel Parkas, Information Systems, Pace--Westchester
Fred Grossman, Information Systems; Doctor of Professional Studies, Pace--New York and White Plains
Fran Goertzel Gustavson, Information Systems, Pace--Westchester
Joseph F. Malerba, Computer Science, Pace--Westchester
John S. Mallozzi, Computer Information Sciences, Iona College
John C. Molluzzo, Information Systems, Pace--New York
Pauline Mosley, Technology Systems, Pace--New York
Narayan S. Murthy, Computer Science, Pace--New York
Catherine Ricardo, Computer Information Sciences, Iona College
Judith E. Sullivan, CSIS Alumna, MS in CS from Pace--Westchester
Sylvester Tuohy, Computer Science, Pace--Westchester

The Ivan G. Seidenberg School of Computer Science and Information Systems, through the Technical Report Series, provides members of the community an opportunity to disseminate the results of their research by publishing monographs, working papers, and tutorials. Technical Reports is a place where scholarly striving is respected.

All preprints and recent reprints are requested and accepted. New manuscripts are read by two members of the editorial board; the editor decides upon publication. Authors, please note that production is Xerographic from your submission. Statements of policy and mission may be found in issues #29 (April 1990) and #34 (September 1990).

Please direct submissions as well as requests for single copies to:

Allen Stix
The Ivan G. Seidenberg School of Computer Science and Information Systems
Goldstein Academic Center
Pace University
861 Bedford Road
Pleasantville, NY 10570-2799