

A Layman's View of Artificial Intelligence

By Daniel P. Glassman

Okay. I've heard enough about what every vendor in the business seems to like to say about their software and all of the artificial intelligence in it. Half the time it sounds like a bunch of teenage boys with rulers arguing over who has....well, never mind. It's just that after a while you get kinda sick of hearing these claims that all claim to match the claims of the other claimant that after a while nobody even knows what the heck they are claiming any more. What also doesn't seem to matter is the total lack of credibility in the specious claims made by so many of these software developers.

It's just too bad that there aren't some clearly defined standards and benchmarks that make it a little easier for the non-computer-propeller-head-type of person to sift through the nonsense floating around in the reporting universe. In the absence of anything too technical, you may want to consider the following as a layman's guide to CAT artificial intelligence.

If a CAT system doesn't know a conjunction from an intersection you can bet that it has a lot of competitive company because few of them do. There is a general collision of the minds amongst software vendors making the claims but not delivering the goods on artificial intelligence.

What is it we are working with here? (Do I really need that question mark there? After all, that's just a rhetorical question?) Reporting has more to do with language than almost any other discipline that I know of short of being a grammar (gramma, here in New England) or foreign-language teacher or a linguist. Steno is simply another language that actually converts to English better through computerized translation than do most other languages. The trick then becomes one of determining which software does the best job of doing that conversion.

To analyze that issue more closely, one must understand what happens during the computerized translation process. There is, of course, the steno input to the software, either via realtime or diskette. That steno must then be processed in some manner to create a text for final editing and printing.

Many CAT software systems use a methodology for this process that evolved out of the late 1970's and early '80's. It is little more than a matching table that matches steno strokes as defined in the user's dictionary to the word, word group or phrase as defined in that dictionary and creates the text by going stroke by stroke through the note file looking for matches. Where matches aren't found, the user ends up having to stop and edit in the text. Or where conflicts are found, the user must select the appropriate conflict or word-group it to have it not appear as a conflict the next time it shows up in that same context. The problem with this basic methodology is that it's great for programmers but it does nothing for reporters in terms of actively analyzing and intelligently creating text, particularly when stroke variations are involved.

Let me shift horses for one moment and explain my view of shorthand writing, something I find rather easy to do since I never had to learn how to do it. But what I observe to be the case in my experience has nothing to do with the mechanics of writing. Rather it has everything to do with the psychology of writing. I have forever flushed the term "misstrokes" from my vocabulary. First of all, it looks goofy with those two s's side-by-side. Secondly, I think that the mental approach to writing is much like the mental approach used for doing anything successfully, whether sports, stuff at home or professional performance. When you approach life as a half-full glass of water you end up with a much more optimistic and positive view of events around you than if your glass is always half empty. So, in my view, misstrokes only exist in school where teachers point out your "errors" because they are measuring your performance in training against a pre-defined standard. Once you leave school, the way you write is the way you write. Your job is to get the proceedings down not concern yourself with how precisely you adhere to the method you learned in school. Everyone's writing evolves over time anyway so rather than thinking negatively, think positively about your writing. If you write a stroke differently, *it's not a misstroke, it's a variation*. It's just another way you write that stroke. Now, teach your software how you write and what some of those more common variations may be, you know, which keys you may more frequently shadow or drop altogether, and your software will be able to accommodate those things in its translation process.

"What?" you say? Tell my software WHAT? Well, that slides me back into the point I was making earlier about CAT software and artificial intelligence. If your software doesn't understand the language in which it's working, doesn't understand how YOU write and goes off and translates some ole' stuff that some programmer predetermined it would do, you have to wonder how smart it really is.

Take Eclipse, for example...rather, buy Eclipse, then you make me a lot happier because I can then put socks on my little children's cold little feet this winter. Let's start with some basics in what the software will do.

First of all you can surprise yourself by obtaining some exceptional translations by defining prefixes and suffixes so that they work with the software to intelligently construct words such as 'enlighten' where the conflict \en^\^en combines with 'light' to translate correctly without having the word 'enlighten' in your dictionary. By defining 'up' as a conflict up^\^up^\^up you can get good translations for words like up, fedup, uptight, upon (POPB)\up on, etc., without having to create other ways of writing them.

Then tell the software what keys on your steno machine you typically shadow (drag) or drop. Good. Your software now knows a little more about how you write. When it translates, if it doesn't find a text match for your steno stroke it will automatically analyze the stroke to determine if it has a shadow that you've told it may sometimes appear or if it has a key missing from a weak finger you've already told it about. If it sees either of these conditions it will remove the shadowed stroke(s) and/or (that's right it can handle both at the same time) insert the strokes you've told it that you drop from time to time and will then go and run a spell check against the word it finds to be sure that it's giving you the correct translation. If it doesn't find a match for the stroke adding or dropping those keys for you the software runs a phonetic translation of the steno, then goes and spell-checks it and you will often end up with correctly translated words that aren't even in your dictionary and that involve neither dropped or dragged keys.

Because we are dealing in language as the main issue, Eclipse also understands the English language like no other CAT software. Which means that, with Eclipse, you are able to write conflicts and variations all you like knowing that the software understands enough about the a) steno, b) how you write and c) the English language so that the translation quality is better than anything another program can do.

What intelligently implemented A/I actually does is to allow you to define things in your dictionary, creating what would for other software be conflicts that you'd have to resolve, so the you can create complexities for the software to sort out rather than you having to write things differently. Take for example the conflicts, \volume\Volume{#R} or \exhibit\Exhibit{#N}. Because the software analyzes the grammatical context of how things appear it will know that when ever a number follows the word volume or the word exhibit, you are referring to a specific Volume or Exhibit and will select the capped option followed by a number that is automatically displayed as a Roman numeral or generic digit respectively.

I'm sure that many people have created different strokes for R E D to differentiate between the verb and the adjective. Well, that's really only necessary if your software doesn't know the difference between a verb and an adjective. True artificial intelligence would be able to understand from the grammatical context (not the word group) in which the stroke appears selecting the correct word. Thus you could write a sentence like, "I read a book on the red bus while going to work in the read building because I heard it was a good read." Writing all of those red\reads as RED would provide you with the correctly written sentence. Go though the list of conflicts that you once learned and try to figure out what you had to do to learn to write them differently. Do you have to momentarily or consciously think of the correct spelling for words or think about how to write the correct word (you know that does nothing but slow you down) when you're writing site\sight\cite, complement\compliment, instep\in step,waive\wave, ... shall I keep going?

What if you've already learned a computer-compatible (= inefficient more complex) theory? No problem. I've never known a reporter who didn't experience an evolution of his or her theory over the years. Inevitably these changes come as newer ways are discovered to write something, combination strokes can be used to (ughhh) resolve conflicts and new shortforms are added. So, how about if you learned a computer-compatible theory and now you find out that your software doesn't need to have you writing everything in different ways. Maybe you could 'evolve' your writing into a more efficient and simplified style that would not only allow you to write faster but more consistently. I've known so many reporters who have told me that they are now (having purchased Eclipse) able to write faster AND they are writing realtime, when on other systems, they never would have.

Like I've said before, I'm no expert on this stuff, but when I keep hearing things like that, there has to be something in it that even I can grasp.

I've been asked about all of the pick-list things that Eclipse does and how much more artificial intelligence there is with that. I'd suggest that the efficiencies brought to Eclipse users through the pick-list things in block files (auto-include/merge docs, etc.) are not EVEN artificial intelligence at all. They are only part of a natural evolution that a

software program under continuous development for over twelve years exhibits. Sophistication in software is rarely a characteristic of developmentally 'young' software. It takes years of work, input from users, implementation of cutting-edge technologies and methodologies, experience in the technology and language of users to develop a program fully. If one's focus is on constantly rewriting a program to get a new look, to create a migration path for all of the disparate users of software a company may have acquired (this is done so they don't have to continue supporting a lot of old programs that they have made obsolete through acquisition, thereby saving money for themselves), or simply to make a product *look* new, the effort gets spent on the rewriting, not the enhancing of the product.

The 'fill-in-the-blank' on Eclipse only scratch the surface of the efficiencies the program provides that are outside the realm of A/I.

A simplified and only partial list of A/I components for you may include the following items. Following the list, we'll talk briefly about some of the advantages these capabilities provide for you because, frankly, if the brilliance of somebody's software is brilliant only for them and doesn't help you in your work, what's the point?

True artificial intelligence:

Understands the language it's working in and gathers intelligence from how you edit to enhance the subsequent translations. Makes intelligent selections from conflicts written by the user based on written language structure. It understands parts of speech, punctuation, colloquy and what should follow it (WITNESS versus A.), and it can identify and distinguish types of sentences (i.e., declarative v. imperative v. interrogative v. exclamatory, etc.) in its decision making models.

Spell checks everything you do making it easier for you to simplify your writing ('er' v. 'or,' etc.)

Learns the way you write and translate your variations accordingly, even to the extent that you get correct translations for strokes not defined in your dictionary..

Understands BOTH languages you work with (steno and text) and provides suggested entries for different ways you may write something in your document.

Understands number usage so that it can make intelligent translations and selections with a minimum of effort from the writer and without requiring you to write different numbers different ways.

Let's talk briefly now on what each of these items means for you. Firstly, any software that requires you to do repetitive tasks over and over again is nothing more than a passive (benign is more like it) interface between you and the computer. If, on the other hand, you select a conflict that teaches the software that a word is an adjective or a noun, the software will be able to make intelligent selections in future situations involving adjectives, and nouns, etc., without necessarily worrying about the specific word itself. The result of this "learning" is that over time, your Eclipse software becomes more and more efficient in the way that it actually helps you do your work without you having to change the way you write your shorthand or without having to group words (which is only useful for those exact words, written precisely the same way, and found in the exact same context).

I've heard some companies claim that they've studied the conflict-resolution issue at great length and have implemented a system for conflict resolution similar to what some voice-recognition companies use. That's great if you're dealing with voice recognition structures. However, written text clearly needs algorithms written for written language. The evidence in comparing the successful implementation of these two approaches leaves me somewhat ambivalent because effective voice-recognition systems haven't had the developmental time to evolve that A/I in CAT systems have. The bottom line for me is that the conflict resolution I've seen in Eclipse is far far superior to anything I've seen from any voice-recognition systems and I've worked extensively with a couple of the more popular ones.

Clearly, understanding a spoken conflict and then trying to determine from the context what the word is and deliver the correct selection is in a different universe from the process used to build a learned intelligence base based on language grammatical structure. Using the logic from one to be effective with the other leads me to believe that a company attempting to mimick voice-recognition methods lacks the knowledge base or experience necessary to do it correctly and is simply attempting to emulate someone else's model for a dissimilar application. It won't work.

Part of a good A/I system design is a process that links all of the functions within the system operation together so that they act and react with some cohesion. Having a contiguous .exe file, for example (versus a bunch of different .exe files strung together to make a program) allows one to have the intelligence of one component work together with another and another to deliver a final translated product that is far superior. It means that when your software has made a selection based on grammatical structure, it can then easily, within the same file execute a spell check, automatically correct the word and

deliver a correctly spelled word. It means that the software can analyze new strokes or variations and after phonetically defining them, spell check them and deliver accurate translations for steno that is heretofore non-existent in your dictionary. It means that you can, with confidence, simplify your writing so that all words ending in either 'er' or 'or' are written as if ending in 'er' saving your OR stroke for the word 'or.' (This is only a single and very basic example that can be expanded greatly to include the '-ible' and '-able' words, '-graph' and '-graphy' words to name just a very few.) It also means that you can get clean translations properly capitalizing words that end in city-name suffixes ('-ville', '-ton', 'bury', 'boro', etc., to get names like Greenville--TKPW ROA E PB/S R EU L--Ivoryton, Sharpton, Canterbury and Millboro) by writing the simple word root you will likely already have in your dictionary with your standard suffix form (i.e., ivory with ton --here you'd be able to create a conflict with \ton^ton\{|-}^ton to get the result you wanted-- sharp with ton, canter with \bury\{|-}^bury, fox with \{|-}^boro, and so on.), etc., without having to have those entries predefined in your dictionary. The description of this capability alone could go on for several pages, so in the interest of saving you all a little snooze time, I'll leave it her letting you use your own imaginations for other possibilities.

When was the last time you had ANY software that was so creatively designed that you could tell it how you operated and have it accommodate your personal style? I've never seen one (until Eclipse, that is) and I've had and worked with a ton of software over the years (both CAT and non-CAT). Typically, they all do what the programmers figure they should do and you're left acclimating to and accommodating the functions built in.

So now, along comes Eclipse with this whole new thing that just happens to link in with the artificial intelligence making it even more effective than it was in the last version. Now you can tell the software what keys are always in the way and what keys you can't always get your fingers to work on. Not only can Eclipse's A/I figure out your own steno based on the dictionary(ies), phonetic translation and spell-checking operations, now it knows when you've written a variation of the way you may ordinarily write. You want to find another CAT system that'll do that for you? Fa'getaboutit! (That's a Brooklyn word.) I'm not even going to start describing all the way that would save you, make you look 'wunnaful' (That's a Hollywood word.), and make realtime writing a breeze for you. If I did, I'd end up with a whole bunch more incredibly long sentences and paragraphs that would confuse the dickens out of some of you, so just kick your imaginations up a notch and imagine what happens when all of that stuff doesn't show up in your text as untranslates. Ahhhh, I thought so.

How many times in a document, a single page or a single paragraph, do you have to define a word you may have written a different way(s)? When you have a program with A/I that understands steno and steno structures, it is able to understand what you're writing and with one definition (seeing what you've written and defined) it can suggest numerous other ways that you might write that word. By adding the suggested variations, you'll find that other ways you've written the word in the document are now resolved and correctly converted to text. On any other CAT system you would have to stop and redefine each different way you may have written the word.

The bottom line in this whole discussion is that as a result of true artificial intelligence, you get much higher quality and more accurate translations as raw (dirty) translations. When coupled with more efficient editing capabilities your ability to write realtime more accurately (without having to change the way you write) and get cleaner translations increases exponentially.

I recently saw some communications on a forum frequented by reporters talking about artificial intelligence. When writing some clarifying comments I received an e-mail back saying "... if you ask all the "top notch" realtime writers they prefer to rely on their implementation of their writing skills rather than a computer software program with S/I." The writer had earlier suggested that she would never rely on A/I for her realtime writing.

Well, it seems to me that suggesting that someone should not rely on A/I for their realtime writing is a bit like suggesting that electric starters are alright for cars but if you drive a bus (making a living from it) you should really be sure you have a hand crank on the front to start the engine.